Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Metodika SIP zátěžových testů

SIP Stress Testing Methodology

2014

Bc. Martin Janota

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Martin Janota**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Metodika SIP zátěžových testů
SIP Stress Testing Methodology

Zásady pro vypracování:

1. SIP a hodnocení výkonnosti SIP prvků pomocí RFC 6076.
2. Aplikace pro generování SIP relací.
3. Vytvoření XML scénářů v sipp pro benchmarking pomocí metod INVITE, REGISTER a OPTIONS.
4. Návrh a realizace generátoru volání s využitím sipp.
5. Benchmarking různých verzí Asterisku pomocí vytvořeného nástroje.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] VOZNAK,M., ROZHON, J. *Approach to stress tests in SIP environment based on marginal analysis.* SPRINGER: Telecommunication Systems, 11p., 2013, ISSN 1018-4864.
[2] VOZNAK,M., REZAC,F., TOMALA, K. *SIP Penetration Test System.* Proceedings The 34th Conference on Telecommunications and Signal Processing, pp. 504-508, August 2010, Baden near Vienna, Austria, ISBN 978-963-88981-0-4.
[3] VOZNAK, M., ROZHON, J. *SIP Back to Back User Benchmarking.* IEEE ICWMC 2010, pp. 92-96, 2010, Valencia, DOI 10.1109/ICWMC.2010.86.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Miroslav Vozňák, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

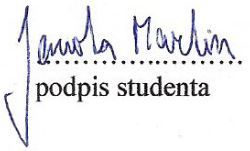
doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *4. května 2014*



podpis studenta

Poděkování

Rád bych poděkoval doc. Ing. Miroslavu Vozňákovi Ph.D. za odbornou pomoc a konzultace při vytváření této diplomové práce a své rodině a blízkým za podporu během doby studia. Dále bych chtěl poděkovat Ing. Janu Rozhonovi za jeho odbornou pomoc a poskytnuté rady při vytváření práce.

Abstrakt

Tato diplomová práce se zabývá hodnocením výkonosti SIP prvků a jejich testováním nejen za pomoci RFC 6076.

Cílem práce bylo vytvořit testovací scénáře a generátor, který ověří vlastnosti Asterisk ústředny. Práce by měla přinést jednotnou metodu testování SIP ústředny s využitím RFC 6076, který není v současné době příliš využíván. Zaměřením na RFC 6076 se práce liší oproti ostatním konkurenčním nástrojům, které přestože generují zátěž a zaznamenávají dobu odezvy, tak nejsou schopny právě s tímto standardem spolupracovat. Dále se práce zabývá testováním jednotlivých verzí Asterisku za účelem jejich porovnání. Z jednotlivých měření by podle RFC 6076 mělo vyplývat, do jakých podmínek je Asterisk schopen plnit časy odpovědí na dané žádosti.

K práci bude využita ústředna Asterisk, generátor testovací zátěže využívající PHP a program SIPp.

Klíčová slova

Asterisk; SIPp; SIP; RFC 6076; Bash;

Abstract

This diploma thesis deal with analysing of performance of SIP elements and theirs tests not only with RFC 6076.

The main aim of work was to create a testing scenarios and generator which will test properties of system based on Asterisk. This work should bring the unified method of SIP panel testing with RFC 6076 which is not mainly used nowadays. Focusing on RFC 6076 is the main difference between this work and others competitive tools. They can generate load tests and write down results as well, but they cannot cooperate with RFC 6076. Another handle of work of this thesis is to test different versions of Asterisk in order to compare them. There should be a clearly result from each measurement based on RFC 6076 to what conditions is Asterisk able to answer on a requests in a specified time.

There will be used Asterisk and generator of load tests using PHP and SIPp in this thesis.

Key words

Asterisk; SIPp; SIP; RFC 6076; Bash;

Seznam použitých symbolů

Symbol	Jednotky	Význam symbolu
t	s	Čas

Seznam použitých zkratek

Zkratka	Význam
SIP	Session Initiation Protocol
VoIP	Voice over Internet Protocol
IETF	The Internet Engineering Task Force
SMTP	Simple Mail Transfer Protocol
HTTP	HyperText Transfer Protocol
MGCP	Media Gateway Control Protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
SDP	Session Description Protocol
IP	Internet Protocol
RFC	Request For Comments
URI	Uniform Resource Identifikator
NAT	Network Address Translation
OS	Operační Systém
PHP	Hypertext Preprocessor
TLS	Transport Layer Security
SCTP	Stream Control Transmission Protocol
PCAP	Packet Capture
RTP	Real-Time Transport Protocol
CSV	Comma-separated values
TDM	Time-division multiplexing
DoS	Denial of Service
GUI	Graphic User Interface
IVR	Interactive Voice Response
API	Application Programming Interface
XML	Extensible Markup Language
B2BUA	Back-to-Back User Agent

Obsah

Úvod

VoIP komunikace nadále čím dále tím více ovlivňuje dění ve světě. Takováto komunikace bývá ve většině případů založena na SIP protokolu. SIP protokol by ovšem měl také splňovat určité parametry pro komunikaci. K tomuto účelu byl organizací IETF zaveden standard RFC 6076. Tento standard zavedl metriky testování právě SIP komunikace. Jakožto stěžejní bod v SIP infrastruktuře je tzv. B2BUA, který SIP komunikaci zprostředkovává a jedná se o nejdůležitější prvek při komunikaci založené na SIP protokolu. Tato práce se soustředí právě na testování B2BUA neboli VoIP ústředny. Pro účely práce byla vybrána ústředna Asterisk, vzhledem k tomu, že je jedna z nejpoužívanějších pro komunikaci založenou na SIP protokolu. K testování VoIP ústředen již existuje řada nástrojů. Mnohé z nich jsou specializovány právě na SIP protokol. Tyto testovací nástroje ovšem ve většině případů jen generují zátěž s tím, že čekají, co daná ústředna vydrží. Veškerá spolupráce s danými standardy chybí. Právě proto vznikl již zmíněný standard RFC 6076 a metriky popsané v tomto standardu jsou využívány v této práci.

Práce se tedy nejdříve zaměřuje na SIP protokol jako takový, s tím, že se jedná o základ celé práce. Díky tomu, že bude monitorován právě SIP provoz, tak je nutné celému procesu vyjednávání spojení mezi SIP zařízeními dopodrobna rozumět. V kapitole 1 jsou popsány jak SIP žádosti, tak SIP zprávy, jakožto zastoupení obou komunikujících stran.

V následující kapitole, kapitole 2, je popsán standard RFC 6076. Tento standard zavedl metriky, jež jsou používány pro testování parametrů SIP komunikace. V této kapitole jsou vysvětleny jednotlivé metriky s názornými obrázky, které zobrazují komunikaci mezi SIP prvky v síti. Z metrik jsou později využívány hlavně dvě nejdůležitější, které nejlépe popisují výkonnost ústředny.[5]

Další kapitola, s číslem 3, se dostává k popisu již existujících generátorů SIP provozu. Postupně jsou popsány čtyři nástroje, které je možné využít. Jsou popsány nástroje jak pro operační systém Windows, tak pro operační systémy založené na Linuxu. Hlavní část kapitoly je věnována programu SIPp[11], který je využíván pro generování provozu v této práci.

Dále jsou vysvětleny již samotné vytvořené scénáře za účelem testování[11]. U každého scénáře je popis toho, jak je scénář přínosný právě pro účely testování a samozřejmě i to, jak jsou scénáře koncipovány po stránce komunikace mezi jednotlivými SIP prvky. U každého scénáře je rovněž uveden čas, případně jiná metrika, která je u scénáře měřena.

Později již následují informace o vytvořeném generátoru. Tento generátor propojuje prvky PHP s knihovnou JpGraph, HTML, nástroje SIPp a skriptů jazyka BASH. Generátor může pracovat se scénáři, které si vytvořil sám uživatel, případně se scénáři připravenými a nastavenými pro otestování za pomoci standardu RFC 6076. V kapitole se rovněž nachází informace o všech instalačních a konfiguračních věcech nutných ke spuštění generátoru a také obsahuje je informace ohledně práce s generátorem. Dále je zde popis topologie, která byla pro testování použita, jak byl nastaven Asterisk a další důležité informace ohledně postupu při

testování. V samotném závěru kapitoly se nachází popis možnosti měření s využitím programu tcpdump i se zdůvodněním nevyužití programu.

Jako poslední je uvedena kapitola zhodnocení dosažených výsledků s porovnáním jednotlivých verzí Asterisku. Jsou popsány jednotlivé testované verze a výsledky, kterých dosáhly při testech. Konkrétně se kromě času, který je základním prvkem metriky RFC 6076, práce zaměřila i na využití procesoru a pamětí během testů.

1 Signalizační protokoly

V současné době existuje několik signalizačních protokolů. Nejpoužívanější zástupci těchto protokolů jsou protokoly MGCP , H.323 a SIP. Tato práce se bude zabývat především protokolem SIP, jehož využívá velké procento uživatelů.[2][20]

1.1 SIP

Session Initiation Protocol (dále jen SIP) je signalizační protokol používaný pro vytváření, správu a ukončení relací v IP sítích. Relace může být jednoduchý dvousměrný telefonní hovor nebo také složitá multimediální konference. SIP nemusí být svázán s žádným jiným protokolem pro přenos multimediálních dat. K vlastní hlavičce SIP zprávy je možné přidat tělo jiné zprávy, které zpravidla popisuje přenášený obsah relace (média). K tomuto účelu je například využíván protokol SDP.[1]

Nepopiratelnou výhodou je jednoduchost. Jedná se o textový protokol, který má podobnou strukturu jako například poštovní protokol SMTP nebo protokolu HTTP. Ačkoliv by z názvu mohlo vyplývat, že se jedná o protokol relační, ve skutečnosti patří SIP mezi protokoly aplikační vrstvy a pracuje nad transportními protokoly UDP či TCP, konkrétně na portu 5060. SIP je neustále se rozvíjející protokol, který vznikl pod záštitou organizace IETF.[2] [7]

1.1.1 Historie SIP protokolu

S nástupem VoIP komunikace bylo nutné vytvořit protokol, který bude tuto komunikaci ovládat a řídit. Do doby než IETF zavedl SIP, bylo vyzkoušeno mnoho návrhů různých protokolů, které se ovšem neuchytily. V roce 1996 tak začaly vznikat základy nového protokolu, který se později začal rozvíjet. V prvopočátcích vzniku v sobě SIP zahrnoval prvky tehdy navrženého protokolu SCIP (Simple Conference Invitation Protocol). Výsledný protokol SIP byl přijat internetovou komunitou v roce 1999 jako RFC 2543 a o rozvoj protokolu se nyní stará IETF a SIP Forum.[4]

Vývoj SIP protokolu[1]:

- Únor 1996 – Počáteční návrhy byly utvořeny ve formě Session Invitation Protocol (vytvořili M. Handley, E. Schoole) a SCIP (navrhnul H. Schulzrinne). SIP byl původně zamýšlen pro vytvoření prostředku, jenž by zval účastníky do velké multimediální konference na internetu tzv. Multicast Backbone (Mbone). V tu chvíli prakticky neexistovala žádná IP telefonie. První návrh byl znám jako "draft-ietf-mmusic-sip-00”. Zahrnoval pouze jeden typ žádosti, a to konkrétně žádost o nastavení hovoru.
- Prosinec 1996 – Novější verze “draft-ietf-mmusic-sip-01” byla navržena jako modifikace první verze. Byl to již první pokus o vytvoření protokolu, jak je znám dnes.

- Leden 1999 – IETF publikovala koncept nazvaný "draft-ietf-mmusic-sip-12". Tento návrh obsahoval již 6 žádostí.
- Březen 1999 – SIP publikován jako standard RFC 2543. Později byl modifikován na novější a modernější verzi RFC 3261 (konkrétně v červnu roku 2002).
- Listopad 2000 – SIP byl přijat jako 3GPP (The 3rd Generation Partnership Project) signalizační protokol a permanentní prvek architektury IIM (IP multimedia subsystem) pro multimediální služby založené na IP protokolu v buňkových systémech.

1.1.2 Funkce SIP protokolu

Funkčně je SIP limitován pouze na nastavení, modifikaci a ukončení relací. Všechny ostatní klíčové funkce musí obstarat jiné protokoly.[1]

SIP má za úkol tyto hlavní věci[1]:

- Umožnit stanovení umístění uživatele (tj. překlad z uživatelského jména na jeho současnou síťovou adresu).
- Stanovit parametry relace tak, aby všichni účastníci konference s těmito podmínkami mohli souhlasit.
- Mechanicky ovládat správu hovorů např. přidávání, ukončování nebo přenesení účastníků.

SIP naopak[4]:

- Neumí zajistit požadovanou kvalitu služby (Quality of Service, QoS). SIP totiž neumí rezervovat síťové prostředky či upřednostnit nějaký provoz. Výhodou je, ale to že, může spolupracovat s protokoly, které QoS mohou zajistit (např. RSVP).
- Přestože jsou http i SIP oba textové protokoly, tak SIP není vhodný k přenosu velkého množství dat například jako již zmíněný http protokol. SIP pro svou práci potřebuje přenášet pouze malý objem dat a také krátké textové zprávy.

1.1.3 Komponenty

SIP je protokol fungující na principu klient-server. Entity spolupracující v SIP protokolu se nazývají User Agent (UA). Existují dva typy UA[1]:

- User Agent Client (UAC, dále jen klient) – Generuje žádosti a zasílá je serveru.
- User Agent Server (UAS, dále jen server) – Přijímá žádosti, zpracovává je a generuje odpovědi.

Jeden UA může pracovat jako oba typy. Kupříkladu telefon funguje jako server pro příchozí hovory a naopak jako klient pro odchozí hovory.

1.1.3.1 Klient

Obecně se pojem klient spojuje s koncovým uživatelem tj. aplikace běžící na nějakém systému používaná lidmi. Může to být softwarový telefon běžící na PC nebo také klasický IP telefon. Klient generuje žádosti např. v případě, že se chce dovolat nějakému dalšímu účastníkovi sítě. Tyto žádosti posílá serveru, který je dále zpracovává[1].

1.1.3.2 Server

Servery jsou hlavní části sítě. Mají předdefinovaný soubor pravidel jak zacházet se žádostmi od klientů. Existuje několik druhů serverů[1]:

- Proxy server – Nejčastější typ serveru v SIP prostředí. Pokud je vygenerována žádost, tak není předem dána adresa příjemce. Klient tím pádem zasílá žádost proxy serveru. Server jménem klienta přepošle žádost na další proxy server nebo je příjemce sám. Proxy server dokáže jak interpretovat žádosti, tak přepisovat hlavičky.
- *Redirect server* – *Redirect server* neboli přesměrovávající server. Jeho hlavní úkol je přesměrovat žádosti od klientů, pokud klient musí zkusit jinou cestu k příjemci. To se obvykle stává, pokud se příjemce přemístil ze své pozice ať už permanentně nebo jen přechodně. Tento server není schopen přijmout ani zpracovat volání.
- *Registrar* – Registrátor zjišťuje aktuální pozici uživatele v síti. Toho je docíleno tím způsobem, že každý příchozí klient se musí registrovat právě u registrátora a také své záznamy u něho pravidelně aktualizovat.
- *Location server* – Adresy registrované v registrátoru jsou uloženy v lokačním serveru.

Servery se také dělí mezi Stateful a Stateless. Stateful server je takový, který si drží stavy transakce a transakce trvají poměrně dlouho, kdežto Stateless server je jednoduchý, rychlý a pouze přeposílá zprávy bez ohledu na vzájemné vazby[8].

1.1.4 Pojmy architektury SIP protokolu

V případě SIP protokolu je možné setkat se s určitými pojmy, které jsou vysvětleny v následujícím seznamu[3][2]:

- SIP adresa – Identifikace uživatele se provádí na základě SIP adresy neboli SIP URI. SIP adresa bývá častokrát až na předponu shodná s e-mailovou adresou, díky svému shodnému formátu, kdy SIP adresa obsahuje navíc jen předponu sip:. SIP adresa může vypadat například následovně: sip:uzivatel@hostitel. Adresa může dále obsahovat i další nepovinné údaje, jako jsou například číslo portu, který pokud není uveden, tak je použit výchozí port 5060. Místo části uživatel může být uvedeno telefonní číslo, případně místo části hostitel může být uvedena IP adresa nebo doménové jméno.

- Transakce – Je to posloupnost SIP zpráv vyměňovaných mezi jednotlivými SIP prvky. Obvykle obsahuje jednu žádost a odpovědi vztažené právě k této žádosti.
- Relace – Relace je částečně permanentní interaktivní výměna informací, také známa jako rozhovor, dialog nebo setkání mezi dvěma nebo více komunikujícími zařízeními případně mezi počítačem a uživatelem.
- Dialog – Jedná se o soubor SIP zpráv, které mají navzájem souvislost. Většinou se jedná o dvě či více transakcí. Například transakce obsahující zprávu INVITE souvisí s transakcí obsahující zprávu BYE a spolu tvoří dialog. K tomu, aby transakce mohly tvořit dialog, musí mít v SIP hlavičce stejná tři pole: *Call-ID, From, To*.
- *End-to-End* – Neboli konec-konec. Princip konec-konec popisuje dva nebo více prvků potřebných pro zahájení vytvoření požadavku, obdržení požadavku a odpovědi na požadavek. Princip také zahrnuje nevyhnutelnou zapojitelnost prvků do dialogu mezi dvěma klienty, servery a interními proxy servery.
- Sestavení relace – K sestavení relace je v SIP protokolu využito tzv. „three-way handshake“. Tento princip je popsán v kapitole 1.1.10.
- Nastavení relace – Jedná se o posloupnost zpráv a s tím zahrnutých parametrů spojených se sestavováním relace s korespondujícím UA.

1.1.5 SIP požadavky

Požadavky jsou základ celého SIP protokolu. Požadavky (také jsou nazývány jako žádosti), zasílá nejčastěji klient na server. V prvním řádku celé SIP zprávy je uveden přesný typ zprávy nebo žádosti, která je právě používána. Existuje několik SIP požadavků. Všechny jsou uvedeny v následujícím seznamu[17]:

- *INVITE* – Žádost *INVITE* slouží k spojení pomocí přizvání uživatele nebo služby. Tělo zprávy obsahuje popis relace (spojení).
- *ACK* – Zpráva *ACK* potvrzuje, že klient v pořádku přijal odpověď na *INVITE* dotaz.
- *BYE* – Zpráva *BYE* se používá k oznámení, že jedna strana hovoru žádá o ukončení spojení. Zpráva *BYE* může být vyslána jak volaným, tak volajícím (tedy jak serverem, tak klientem)
- *CANCEL* – Zpráva *CANCEL* ukončuje nevyřízenou žádost *INVITE*. Nevyřízená žádost, jež má být ukončena musí mít v SIP hlavičce stejnou identifikací, to znamená položky *Call-ID*, *To*, *From* a *CSeq*.
- *REGISTER* – Žádost *REGISTER* je využívána k registraci klienta u registračního serveru. Registrace je nutno obnovovat, neboť jsou časově omezeny.
- *OPTIONS* – Žádost pro zjištění informací ohledně funkcí podporovaných serverem.
- *INFO* – Zpráva *INFO* slouží k zasílání informací, které nemohou ovlivnit stav relace za chodu.
- *UPDATE* – Zpráva je určena ke změně informací za chodu, aniž by byly ovlivněny parametry dialogu.

- *PRACK* – Dočasné potvrzení.
- *SUBSCRIBE* – *SUBSCRIBE* žádost má za úkol oznamovat příjem/odběr události či zprávy.
- *NOTIFY* – Zpráva využívaná k informování odběratele o nové události.
- *MESSAGE* – Přenos rychlých zpráv pomocí SIP protokolu.

1.1.6 Hlavička SIP žádosti

SIP zprávy, ať už se jedná o žádosti či odpovědi na ně, mají hlavičky, které danou zprávu specifikují. V této kapitole bude představena hlavička SIP žádosti[1]. Tyto žádosti jsou ve většině případů vytvářeny klienty a dále jsou pak zasílány na server. Na obrázku 1.1 lze vidět, jak taková žádost může vypadat, konkrétně se jedná o žádost INVITE.

```
INVITE sip:user2@server2.com SIP/2.0
Via: SIP/2.0/UDP pc.server1.com;branch=z9hG4bK776asdhds
To: user2 <sip:user2@server2.com>
From: user1 <sip:user1@server1.com>;tag=1928301774
Call-ID: 11-25413@pc.server1.com
CSeq: 314159 INVITE
Contact: <sip:user2@pc.server2.com>
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: 112
```

Obr. 1.1: *Hlavička SIP žádosti*

Jak je patrné z obrázku 1.1, tak se hlavička skládá z několika částí. Následně budou tyto části zkráceně popsány.

- *Request-line* – První řádek je nazýván *Request-line* a je složen z několika částí. Na první pozici řádku se nachází název použité žádosti, v případě z obrázku 1.1 se konktrétně jedná o žádost *INVITE*. Za typem žádosti se nachází URI neboli jednotný identifikátor zdroje. Jako poslední parametr je uvedena verze SIP protokolu.
- *Via* – Naznačuje cestu zprávy. Jakýkoliv proxy server na cestě zprávy, zde vkládá své údaje. Pokud zpráva směřuje zpět, tedy je posílána odpověď, jsou tyto údaje proxy serverem odebírány.
- *Max-Forwards* – Počet přeskoků, kterými může žádost přejít, než se dostane ke koncovému uživateli.
- *To* – Adresa koncového uživatele. Skládá se ze zobrazovaného jména a URI adresy.
- *From* – Adresa odesílatele. Je ve stejném tvaru jako adresa koncového uživatele.
- *Call-ID* – Jedná se o globální unikátní identifikátor hovoru.
- *CSeq* – Obsahuje celé číslo a název metody.

- *Contact* – Skládá se ze SIP URI, což je přímá cesta k odesílateli, dále z uživatelského jména a plného doménového jména (FDQN). Může obsahovat také IP adresu.
- *Content-Type* – Obsahuje popis těla zprávy.
- *Content-Lenght* – Oktet počtu těla zprávy.

Hlavička může mít i další nepovinná pole.

1.1.7 Hlavička SIP odpovědi

Každá žádost, pokud je doručena k příjemci, musí mít také odpověď. Odpovědi mají podobný tvar jako žádosti. Obsahují rovněž hlavičku, ve které jsou specifické údaje[1]. Jedna odpověď (klasická SIP zpráva) je zobrazena na obrázku 1.2.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.30:5060;received=192.168.1.31
To: user2 <sip:user2@server2.com>;tag=794fe65c1
From: user1 <sip:user1@server1.com>;tag=1928301774
Call-ID: 11-25413 @192.168.1.30
CSeq: 314159 INVITE
Contact: <sip:user2@192.168.1.30>
Content-Type: application/sdp
Content-Length: 112
```

Obr. 1.2: *Hlavička SIP odpovědi*

Hlavička SIP odpovědi se skládá z těchto hlavních částí:

- *Status-line* – První řádek odpovědi se nazývá *Status-line*. První část řádku nese verzi SIP protokolu. Další část je tzv. Status kód, což je kód odpovědi. Poslední úsek odpovědi obstarává důvodová fráze.
- *Via* – *Via* je část hlavičky, která se může se vyskytnout i více než jednou. To proto, že každá komponenta sítě, přes kterou projde žádost *INVITE,* přidá do odpovědi svou identitu. Postupně po dobu procházení přes proxy servery jsou tyto *Via* části hlaviček odebírány každým proxy serverem, než se odpověď dostane ke koncovému bodu (původci žádosti).
- *To* – Identifikace toho komu je odpověď určena, neboli identifikace volajícího.
- *Contact* – Přesná identifikace klienta, kterému byla žádost určena.

Všechny ostatní hodnoty mají naprosto stejný význam jako v předchozím případě, kapitole 1.1.6.

1.1.8 Typy SIP odpovědí

SIP odpověď se skládá ze tří číslic. První číslice definuje kategorii odpovědi. Jakákoliv odpověď 100-199 tedy znamená odpověď z kategorie „1XX“. SIP/2.0 zná šest kategorií odpovědí. Jsou velmi podobné HTTP protokolu. Za třemi číslicemi je uvedena důvodová fráze,

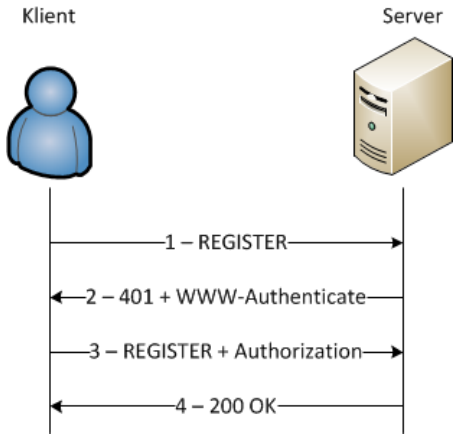
která krátce značí, co daný kód znamená[1]. V následujícím seznamu jsou vyznačeny všechny kategorie odpovědí[6]:

- 1XX – Prozatímní odpovědi, jako požadavek přijat, vyzvání.
- 2XX – Úspěch. Požadavek je přijat, pochopen a akceptován.
- 3XX – Přesměrování. Je třeba vytvořit nový upravený požadavek.
- 4XX – Chyba klienta. Špatná syntaxe požadavku, nebo požadavek nemůže být proveden.
- 5XX – Chyba serveru. Server není schopen provést platný požadavek.
- 6XX – Globální chyba. Požadavek nelze provést na žádném serveru.

Odpovědi s kódem 200 a výše jsou konečné a jejich přijetí ukončuje transakci.

1.1.9 Registrace

Vzhledem k nutnosti ověřování identity klientů, obsahuje SIP protokol žádost *REGISTER*. Registrační proces se skládá z několika po sobě jdoucích úkonů. Nejprve musí klient o samotnou registraci zažádat u registračního serveru. Poté je ze serveru poslána odpověď o nutné autentizaci, s informacemi ohledně autentizace. Klient tedy v další žádosti zašle potřebná data, které po ověření serverem skončí úspěšnou registrací na serveru. Registraci je nutno obnovovat, neboť je časově omezena. Na obrázku 1.3[18] je možné vidět postup registrace klienta u registračního serveru.

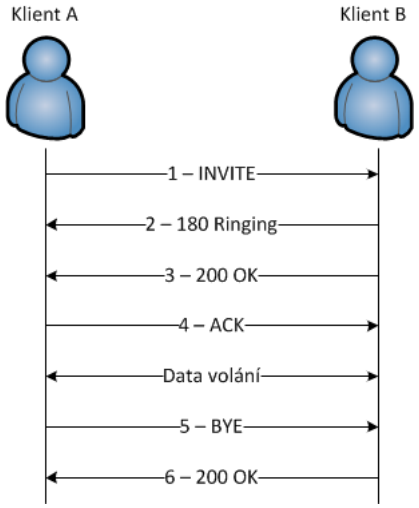


Obr. 1.3: *Registrace*

1.1.10 Navázání a ukončení spojení

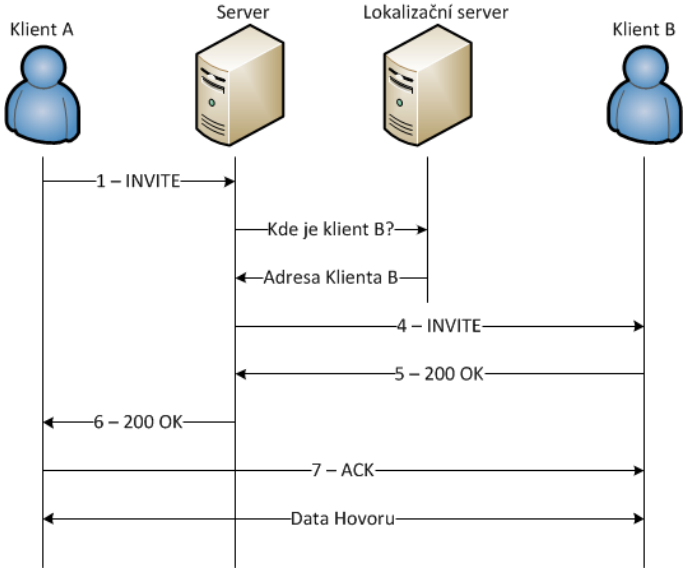
Spojení je u SIP protokolu řešeno pomocí takzvaného „*three-way handshake*“. Pokud chtějí dva klienti mezi sebou navázat kontakt, musí se řídit následujícími kroky. Klient A, jenž chce vytvořit spojení s klientem B, musí nejprve poslat žádost *INVITE*. V této žádosti se mimo jiné nachází i popis spojení. Pokud žádost dorazí k volanému klientovi, pro příklad Klient B, tak volaný klient, pokud žádost nezamítne rovnou z jakéhokoliv důvodu, posílá zpět Klientovi A zprávu, že hovor akceptoval a je připraven k hovoru například posláním zprávy *180 Ringing*.

Jestliže chce Klient B hovor přijmout, tak Klientovi A posílá zprávu *200 OK*. Odpověď obsahuje data informující Klienta A, kde volaný účastník, Klient B, očekává data od volajícího klienta, informace o IP adrese, kodeku atd. Po přijetí zprávy *200 OK* zasílá Klient A žádost *ACK*, čímž potvrdí volání a konverzace mezi klienty může začít. Pro ukončení hovoru je nutné, aby jedna z komunikujících stran zaslala žádost *BYE*, která pokud je potvrzena zprávou *200 OK*, ukončí hovor. Obrázek 1.4 znázorňuje právě takovouto situaci[2].



Obr. 1.4: *Three-Way Handshake*

V předchozím případě je popisováno spojení mezi dvěma klienty, kteří znají své adresy. Tato situace, ale v reálném systému nastává jen velice zřídka. Klienti se musí registrovat u lokalizačních serverů, které znají přesné adresy všech klientů v síti a díky nim mohou klienti komunikovat. Na dalším případu bude ukázán přesně tento případ, kdy Klient A chce komunikovat s Klientem B, ale neznají své přesné adresy. Volající, tedy Klient A, zašle žádost *INVITE* na server. V této žádosti musí být jasně specifikováno to, s kým má být započnutý hovor. Server dále zažádá lokalizační server o zjištění adresy Klienta B. Po získání této informace dále server již posílá žádost *INVITE* Klientovi B, pokud se server nachází v proxy módu. Pokud je v takzvaném „*redirect*“ módu, tak žádost pošle zpět klientovi, na jehož uvážení je poté s žádostí nakládáno. Nadále bude popisován proxy mód. Tedy po zaslání žádosti Klientovi B nastává podobný postup jako u předchozího příkladu, s tím, že pro zjednodušení tohoto příkladu je vynechána část *180 Ringing*. Po doručení zprávy *200 OK* zpět na volající stranu, Klienta A, tak již komunikují navzájem klienti mezi sebou[2]. Tento příklad nastává tehdy, pokud není použit mechanismus záznamu směrování, který je popsán v kapitole 1.1.12.



Obr. 1.5: *Three-Way Handshake s lokalizací*

Ukončení hovoru z obrázku 1.5 probíhá stejným způsobem jako v předchozím příkladu.

1.1.11 Směrování SIP zpráv

SIP protokol využívá pro směrování položky v hlavičce zprávy. Jedná se konkrétně o adresu *Request-URI* a položky *Via*. Prvek, kterým daná žádost prochází, umístí vždy svou adresu právě do položky *Via*. Udělá to takovým způsobem, že svou adresu umístí vždy před ostatní Via položky v hlavičce, pokud už hlavička nějaké obsahuje. Naopak do adresy *Request-URI* jsou zapisováni další příjemci požadavku. Konkrétně prvek, který přijme zprávu, vloží do adresy *Request-URI* adresu dalšího příjemce. Adresa *Request-URI* obsahuje vždy jen jednu adresu, kdežto položek *Via*, může být i více. Při rozesílání odpovědi musí prvek, který odpovídá, zkopírovat všechny položky *Via* do odpovědi. Po provedení tohoto kroku posílá prvek odpověď na adresu uvedenou v první položce *Via*. Příjemce této odpovědi tuto první položku *Via* odstraní, neboť to byla jeho vlastní adresa a odpověď posílá na další uvedenou položku *Via*.[2]

1.1.12 Mechanismus záznamu směrování

Ačkoliv je možné, aby klienti komunikovali v rámci hovoru vzájemně mezi sebou, tak je někdy nutné, aby proxy server sledoval všechny zprávy putující mezi komunikujícími klienty. Například pokud proxy server musí zajišťovat účtování, tak tím pádem musí vědět, kdy je zaslána zpráva *BYE*. K tomuto určený mechanismus se nazývá mechanismus se záznamem směrování. Takto proxy server sděluje klientům, že si přeje dostávat všechny informace, které si klienti vyměňují. Aby toho docílil, tak vloží do hlavičky SIP zprávy pole *Record-Route*. V této hlavičce je zapsána adresa serveru, tak aby klienti věděli, přes který server mají své zprávy posílat. Jakmile příjemce zprávy obdrží zprávu s touto hlavičkou, tak je povinen ji zařadit do své zprávy odesilateli také, aby i odesilatel věděl, kde má své zprávy posílat.[2][8].

2 RFC 6076

RFC 6076 je základní výkonová metrika pro princip konec-konec v SIP telefonii. Konkrétně se jedná o standardní set běžných metrik, dovolující například interoperabilní výkonové měření. V této době, kdy je SIP široce používaným standardem používaným mnohými poskytovateli, koncovými uživateli v telekomunikačních sítích atd., je nutné, aby existovala nějaká výkonnostní metrika právě pro tento protokol. Přestože existuje mnoho různých standardů pro měření výkonu telefonních signalizačních protokolů, takových jako Signaling System 7 (SS7), tak na poli výkonnostního testování a měření SIP protokolu nebyla, až do příchodu RFC 6076, žádná metrika specializující se jen na SIP protokol. Tento standard sice vznikl teprve nedávno, ale vzhledem k rostoucímu počtu instalací VoIP infrastruktury založené na SIP protokolu nabývá tento standard čím dále tím více na důležitosti.[10][9]

2.1 Časový interval měření

Jako skoro v každé metrice i v RFC 6076 je nutné měřit čas. Konkrétně se jedná o měření času mezi dvěma událostmi. V následující kapitole jsou popsány termíny, které s měřením času této metriky souvisí.[5]

t1 – počáteční čas

Je to okamžitá doba (kdy je vyslán požadavek), která začíná kontinuální časový interval. Čas t1 je měřen v době, kdy je žádost zpracována SIP aplikací a první bit paketu žádosti je poslán směrem od klienta nebo proxy serveru.

t4 – konečný čas

Čas, který uzavírá nepřetržitý časový interval po odeslání související žádosti (takové, po které byl spuštěn čas t1). Čas t4 je měřen v době, kdy je přijat poslední bit paketu související SIP žádosti aplikací na klientovi nebo proxy serveru, který žádost odeslal.

Časy t2 a t3 jsou rezervovány pro možné budoucí využití na jiném rozhraní podílejícím se na odpovědi.

2.2 Metriky výkonu SIP

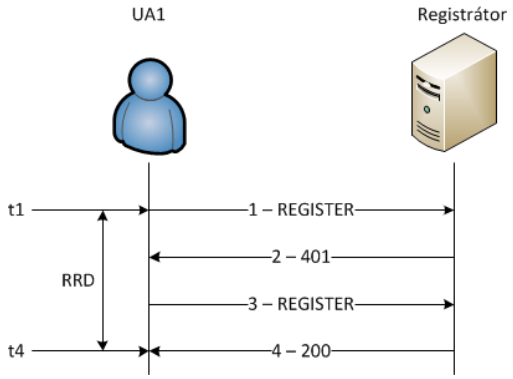
Ve všech následně zmíněných metrikách začíná čas t1 s první zprávou zaslanou klientem. Tento čas není resetován ani, když klient musí poslat stejnou zprávu, v rámci stejné transakce, vícekrát. Konečný čas je založen na konečné správné odpovědi, či na konečné chybné odpovědi. S ohledem na všechny metriky se přesnost a rozlišovací schopnost výstupních hodnot vztahuje k přesnosti a rozlišovací schopnosti vstupních hodnot. Ačkoliv není specifikována velikost vzorku, je nutné tento údaj vzít v úvahu. S větším počtem vzorků mohou metriky přinést přesnější údaje.[5]

2.2.1 Registration Request Delay (RRD)

Registration Request Delay (Zpoždění při žádosti o registraci, dále jen RRD)[5], je doba zpoždění, která vzniká při odpovědi na žádost o registraci od UA. Čas RRD musí být měřen a zapsán vždy pouze při úspěšné registraci, zatímco při neúspěšném pokusu o registraci je počítána metrika *Ineffective Registration Attempts* (popsána v následující sekci). Výstupní hodnota této metriky je číselná a měla by být uvedena v milisekundách. Pro výpočet se používá následující vzorec:

$$RRD = \check{C}as\ konečné\ odpovědi - \check{C}as\ zaslání\ žádosti\ o\ registraci \qquad (2.1)$$

Další obrázek je uveden jako příklad k zobrazení všech událostí nezbytných pro úspěšné dokončení registrace a tím pro možnost výpočtu RRD:



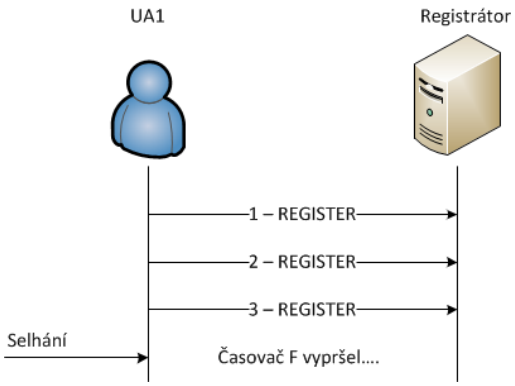
Obr. 2.1: *RRD*

2.2.2 Ineffective Registration Attempts (IRA)

Ineffective registration attempts (Neúspěšné pokusy o registraci, dále jen IRA) je metrika vytvořená pro monitorování neúspěšného pokusu o registraci nebo neschopnost registrátora obdržet žádost o registraci ze strany UA.[5] Tato metrika je měřena na UA začínajícím transakci. Hodnota této metriky je numerická a měla by být označena jako procento neúspěšných registračních pokusů. Výpočty IRA procent jsou prováděny podle následujícího vzorce:

$$IRA\ \% = \frac{\#\ IRA}{\#\ žádostí\ o\ registraci} * 100 \qquad (2.2)$$

Neúspěšná žádost o registraci je definována jako konečné selhání odpovědi na žádost o registraci. To obvykle signalizuje poruchu obdrženou od cílového registrátoru, interního proxy serveru či z důvodu vypršení časového limitu registrace. Chybná odpověď je popsána zprávou 4XX (vyjma zpráv 401, 402 a 407), 5XX nebo 6XX. Vypršení časového limitu je signalizováno uběhnutím časovače F.



Obr. 2.2: *IRA*

Na následujícím obrázku jsou znázorněny události nutné k vypršení časového limitu pro registraci vedoucí ke vzniku IRA. Je možné vidět, že se UA několikrát snaží poslat žádost o registraci na registrátor dříve než vyprší časovač F. Pro kalkulaci IRA je ovšem důležitý pouze první pokus. Další žádosti jsou identifikovány jako další pokusy ve stejné transakci a tím pádem nejsou započítávány. Konečný čas metriky bývá, jak již bylo zmíněno, často spojen i s chybovou zprávou přijatou od registrátoru. Pokud by tedy došlo k mírné úpravě situace z obrázku 2.2, tak by selhání časovače bylo nahrazeno zprávou např. 503, a selhání by bylo počítáno jen do té doby.

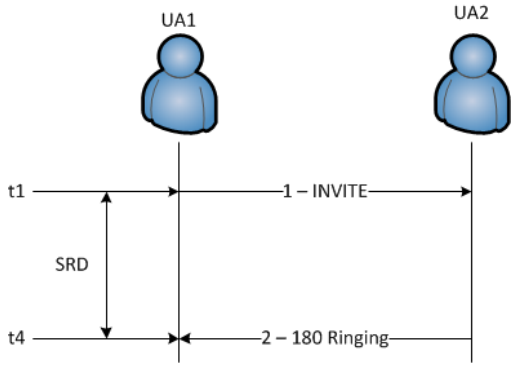
2.2.3 Session Request Delay (SRD)

Session Request Delay (Zpoždění při žádosti o relaci, dále jen SRD)[5] se používá pro detekci chyb nebo znehodnocení způsobující zpoždění v odpovědi UA na žádost o relaci. SRD se měří jak pro úspěšné, tak pro neúspěšné sestavení relace. Doba trvání při úspěšných pokusech by měla být pravděpodobně výrazně jiná nežli doba trvání při neúspěšných pokusech. Výstupní hodnota této metriky musí indikovat, zda je výstup úspěšný nebo neúspěšný a měla by být uvedena v jednotkách sekund. Pro výpočet SRD je použit následující vzorec:

$$SRD = \text{Čas statusu oznámení odpovědi} - \text{Čas pozvání do relace} \qquad (2.3)$$

2.2.3.1 Úspěšné sestavení relace SRD

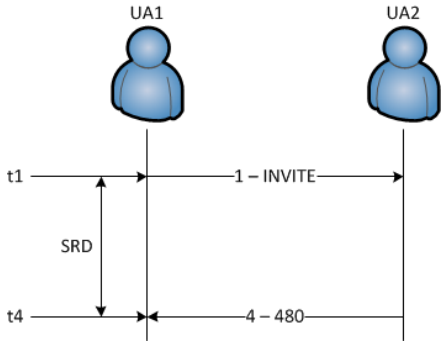
Při úspěšné žádosti je SRD definován jako časový interval od prvního bitu *INVITE* zprávy obsahující potřebné informace zaslané počátečním UA, dokud není přijat poslední bit první předběžné odpovědi.[5] Pro výpočet času SRD při úspěšném sestavení relace jsou používány zprávy 1XX. Ovšem pokud žádná z těchto zpráv není součástí relace, je možné využít pro měření i zprávu 200 OK. Na následujícím obrázku jsou popsány všechny nezbytné události pro úspěšné sestavení relace:



Obr. 2.3: *Úspěšné SRD*

2.2.3.2 Neúspěšné sestavení relace SRD

V případě neúspěšné žádosti je SRD definován jako časový interval od doby, kdy je první bit inicializační zprávy *INVITE* zaslán původním UA nebo uživatelem na konečné zpracování či na koncové UA, až do posledního bitu první prozatímní odpovědi nebo do indikace selhání.[5] Selhání je definováno jako zpráva 4XX (výjimky tvoří zprávy 401, 402, 407), 5XX nebo 6XX. Změny v metrice výstupu můžou indikovat problémy v navazování signálních funkcí, které mohou zneškodnit doručení zprávy *INVITE* k zamýšlenému UA nebo mohou znamenat problém na koncovém zařízení. Na následujícím obrázku jsou popsány všechny nezbytné události pro výpočet SRD při neúspěšné žádosti o sestavení relace:



Obr. 2.4: *Neúspěšné SRD*

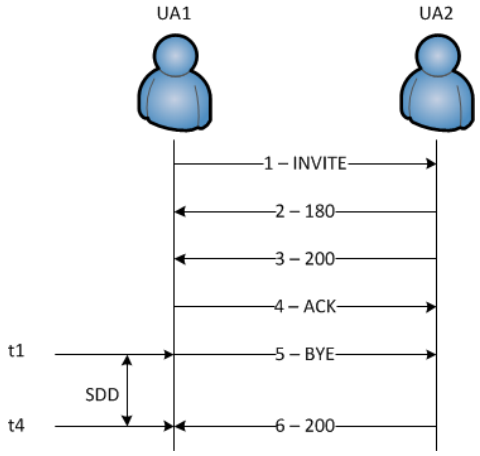
2.2.4 Session Disconnect Delay (SDD)

Session Disconnect Delay (Zpoždění při ukončení relace, dále jen SDD) je metrika, jenž se používá k detekci chyb nebo ke zpoždění při ukončování relace.[5] Čas SDD je měřen jak pro úspěšné ukončení relace, tak pro neúspěšné ukončení relace. Čas SDD může býti měřen jak na zdrojovém UA, tak na koncovém UA, podílejícím se na SIP dialogu. Výstupní hodnota tohoto parametru je numerická a měla by být uváděna v milisekundách. SDD je vypočítáván podle následujícího vzorce:

$$SDD = \text{Čas zprávy } 2XX \text{ nebo Vypršení časového intervalu} - \text{Čas zprávy } BYE \quad (2.4)$$

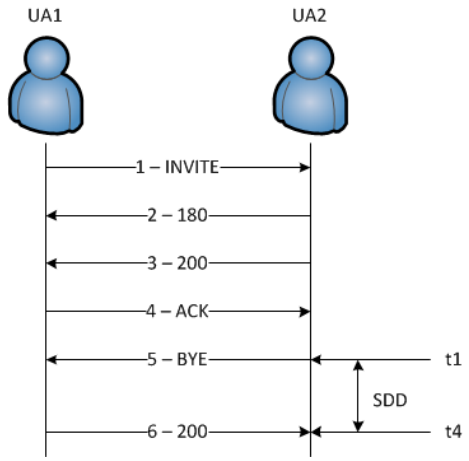
Hodnota metriky SDD je definována jako interval mezi prvním bitem zaslané zprávy informující o ukončení relace, např. *BYE,* a posledním bitem následně přijaté odpovědi, zprávy 2XX. V některých případech je možný příjem chybné opravitelné odpovědi, například 503. V těchto situacích nelze tuto odpověď použít pro výpočet SDD. Místo toho je použita úspěšná odpověď 2XX spojená s obnovením zprávy. Následující obrázek znázorňuje příklad nutných událostí pro kalkulaci SDD při úspěšném ukončení relace:

Měření SDD na počáteční UA (UA1):



Obr. 2.5: *SDD počáteční UA*

Měření SDD na cílovém UA (UA2):



Obr. 2.6: *SDD cílové UA*

2.2.5 Session Duration Time (SDT)

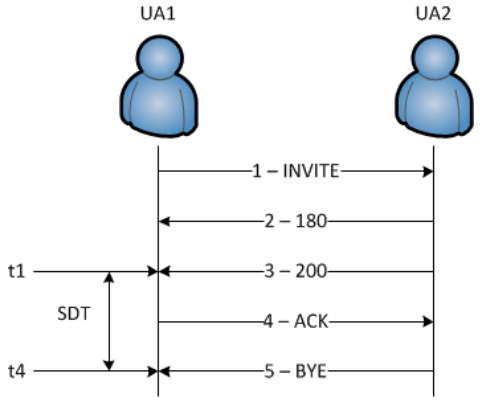
Session Duration Time (Doba trvání relace, dále jen SDT) je metrika, která se používá pro detekci problémů (např. špatná kvalita zvuku), způsobených krátkým trváním relace.[5] Čas SDT je měřen jak pro neúspěšné, tak pro úspěšné relace. Čas SDT může být měřen na jakémkoliv koncovém UA v SIP dialogu, jakožto na počátečním UA v SIP dialogu. Výstupní hodnota této metriky je numerická a měla by být uváděna v sekundách. SDT se počítá podle následujícího vzorce:

$$SDT = \check{C}as\ zprávy\ BYE\ nebo\ Vypršení\ času - Odpověď\ 200\ OK\ na\ zprávu\ INVITE \quad (2.5)$$

2.2.5.1 Úspěšné dokončení relace SDT

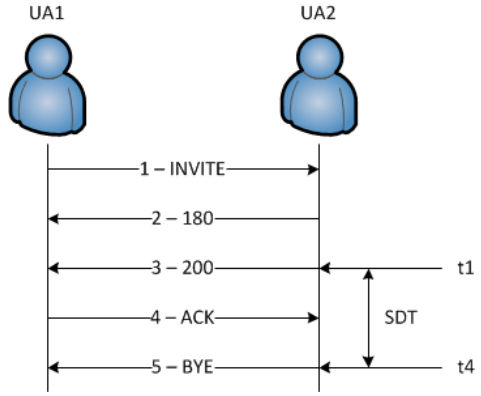
Při úspěšném dokončení relace *(Successful Session Duration)* je SDT počítán jako průměr a je definován jako doba trvání dialogu daného intervalem mezi přijetím prvního bitu zprávy *200 OK* jako odpovědi na zprávu *INVITE* a přijetím posledního bitu asociované zprávy *BYE* označující ukončení dialogu.[5] Opakovaný přenos zpráv *200* a *ACK* kvůli výpadkům spojení neobnovuje časovač této metriky. Následující obrázek ukazuje důležité události nutné pro to, aby mohl být vypočítán SDT při úspěšném dokončení relace:

Měření SDT na původním UA (UA1):



Obr. 2.7: *Úspěšné SDT na původním UA*

Pokud je měření prováděno na koncovém UA (v přechozím případě by to znamenalo UA2), tak SDT je interval mezi zasláním prvního bitu zprávy *200 OK* jako odpovědi na zprávu *INVITE* a obdržením posledního bitu asociované *BYE* zprávy. Pro lepší ilustraci je opět k dispozici obrázek, jenž vše popisuje:

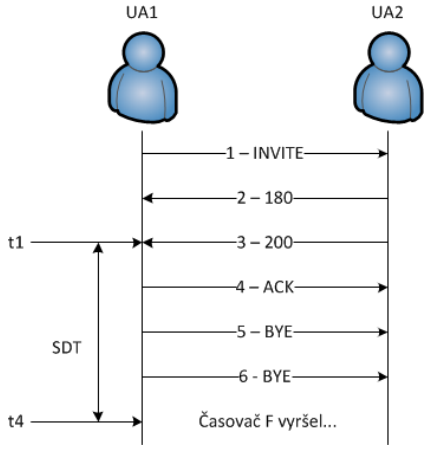


Obr. 2.8: *Úspěšné SDT na cílovém UA*

2.2.5.2 Neúspěšné dokončení relace SDT

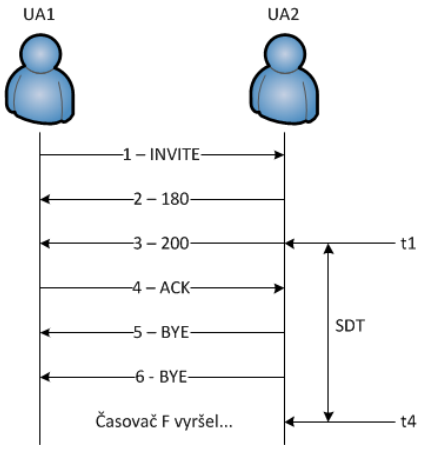
V některých případech není přijata žádná odpověď poté, co je zaslána zpráva o ukončení relace.[5] V těchto případech je SDT definován jako interval mezi obdržením prvního bitu zprávy *200 OK* jako odpovědí na zprávu *INVITE*, a časem kdy vyprší časovač F. Další obrázek vše znázorňuje:

Měření SDT na původním UA (UA1) :



Obr. 2.9: *Neúspěšné SDT na původním UA*

Pokud je SDT počítáno na UA2, tak je definováno jako interval mezi zasláním prvního bitu zprávy *200 OK* jako odpovědi na zprávu *INVITE* a vypršením časovače F. Pro ukázku je přiložen následující obrázek:



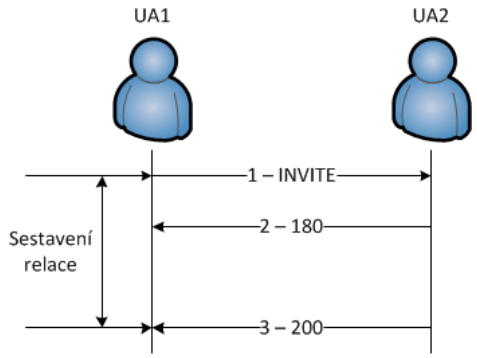
Obr. 2.10: *Neúspěšné SDT na cílovém UA*

2.2.6 Session Establishment Ratio (SER)

Session Establishment Ratio (Poměr založení relace, dále jen SER) je metrika používaná pro detekci schopnosti sestavení nové relace u UA či proxy serveru, jež právě ukončil nebo ukončuje jinou relaci.[5] SER je definován jako poměr počtu *INVITE* žádostí o novou relaci, jejichž výsledkem je zpráva *200 OK*, k celkovému počtu *INVITE* žádostí méně *INVITE* žádosti končící zprávou *3XX*. Tato metrika je měřena pouze na počátečním UA. Výstupní hodnota je numerická a měla by být uváděna tak, aby vyjadřovala procento úspěšných zavedení relace. SER je počítána podle následujícího vzorce:

$$SER = \frac{Počet\ žádostí\ INVITE\ končících\ zprávou\ 200\ OK}{(Počet\ INVITE\ žádostí - INVITE\ končící\ zprávou\ 3XX)} * 100 \qquad (2.6)$$

Příští obrázek popisuje události nezbytné pro sestavení relace:



Obr. 2.11: *SER*

2.2.7 Session Establishment Effectiveness Ratio (SEER)

Session Establishment Effectiveness Ratio (Účinný poměr založení relace, dále jen SEER)[3][5], jedná se o metriku podobnou té, která byla popsána v předchozí kapitole. Tato

metrika má ovšem za cíl vyloučit dopady na koncového uživatele jednotlivých UA. Metrika je definována jako poměr počtu žádostí *INVITE* končících odpovědí *200 OK* a žádostí *INVITE* končících odpověďmi *480, 486, 600* nebo *603* (dále uváděny jako zprávy SEER); k celkovému počtu *INVITE* žádostí méně *INVITE* žádosti končící odpovědí *3XX*. Kódy odpovědí zpráv SEER byly zvoleny proto, že indikují vliv jednotlivých uživatelů UA. Je totiž možné, že někteří jednotliví uživatelé mají negativní vliv na činnost UA. Metrika je měřena pouze na počátečním UA. Výstupní hodnota by měla vyjadřovat procentuální úspěšných sestavení spojení méně běžné UAS chyby. SEER je počítán podle následujícího vzorce:

$$SEER = \frac{Počet\ INVITE\ žádostí + INVITE\ končící\ zprávou\ SEER}{(Počet\ INVITE\ žádostí - INVITE\ končící\ zprávou\ 3XX)} * 100 \qquad (2.7)$$

Referenční příklady jsou uvedeny v kapitole SER (kapitola 2.2.6).

2.2.8 Ineffective Session Attempts (ISAs)

K *Ineffective Session Attempts* (Neúspěšné založení relace, dále jen ISAs) dochází v případě pokud proxy server nebo UA interně vytvoří sestavovací požadavek se špatnými podmínkami[3][5]. Tato metrika by měla indikovat procento neúspěšných sestavení relace. Odpovědi představující tuto metriku jsou: 408, 500, 503, 504. ISAs je počítána jako procento z celkového počtu žádostí, přesněji vyjádřeno v následujícím vzorci:

$$ISAs\ \% = \frac{Počet\ ISAs}{Celkový\ počet\ žádostí} * 100 \qquad (2.8)$$

2.2.9 Session Completion Ratio (SCR)

Session Completion Ratio (Poměr úspěšného dokončení relace, dále jen SCR).[5] Úspěšné dokončení relace je definováno jako SIP dialog, který skončí, aniž by se v něm objevila chyba z nedostatku odpovědí ze strany proxy serveru či UA. Výstupní hodnota této metriky by měla indikovat procento úspěšných dokončení relace. Je počítána podle následujícího vzorce:

$$SCR\ \% = \frac{Počet\ úspěšných\ dokončení\ relace}{Celkový\ počet\ žádostí\ o\ relace} * 100 \qquad (2.9)$$

3 Aplikace pro generování SIP relací

Aplikací pro generování SIP relací existuje velké množství. Drtivá většina z těchto programů umí nejen vygenerovat zátěž, ale také zaznamenávat statistiky. Druhy zaznamenávaných statistik se liší program od programu. Ovšem možnost zaznamenávat tyto statistiky podle nějakého standardu těmto nástrojům chybí. Proto je nutné, aby některý z těchto programů byl doplněn o nadstavbu, která toto umožní. Následuje několik běžně používaných generátoru SIP relací s jejich popisem.

3.1 SIPp

SIPp je nástroj pro testování výkonu SIP protokolu, který zahrnuje několik základních scénářů (UAC, UAS a další), jakožto možnost vytvářet si i vlastní scénáře. Nástroj umí vytvářet velké množství volání se všemožnými SIP žádostmi, tím ověřuje vlastnosti testovaného systému. Program dále obsahuje dynamický displej zobrazující informace o právě běžícím testu. Mezi tyto informace patří například statistiky ohledně přijatých zpráv, míra hovorů či informace o využitém TCP/UDP spojení.[11][21]

3.1.1 Licence

Celý projekt SIPp je vázán standardní GNU GPL licencí. Program byl původně vytvořen a propagován do SIP komunity firmou Hewlett-Packard, avšak momentálně není touto firmou poskytována tomuto programu žádná podpora.[11]

3.1.2 Dostupné platformy

Nástroj SIPp je možný spustit na skoro všech platformách UNIX: HPUX, Tru64, Linux (RedHat, Debian, FreeBSD), Solaris/SunOS. Spustitelný je rovněž na OS Windows, kde je ovšem nejdříve nutné SIPp sestavit pomocí programu Cygwin. Protože SIPp plně podporuje IPv6, tak je ho možné spustit jen na Windows XP a novějších OS Windows. Díky tomu, že se práce soustředila na využití nástroje na operačním systému Ubuntu 12.04, tak dále budou popisovány jen postupy, pro tento operační systém.[11]

3.1.3 Instalace

Na Linuxu je SIPp poskytován ve formě zdrojového kódu.[11] Proto je nejdříve nutné jej sestavit. Nutné prvky pro sestavení jsou následující:

- C++ sestavovač
- knihovny *curses* nebo *ncurses*
- pro TLS podporu: OpenSSL >= 0.9.8
- pro podporu přehrávání pcap: knihovny *libpcap* a *libnet*
- pro SCTP podporu: *lksctp-tools*

Existují 4 možnosti toho jak SIPp sestavit. Nicméně všechny možnosti mají prakticky stejný průběh a liší se jen v jednom příkazu při zadávání do příkazového řádku. Pro účely práce je důležitá pouze jedna možnost sestavení:

```
tar -xvzf sipp-xxx.tar
cd sipp
autoreconf -ivf
./configure --with-sctp --with-pcap --with-openssl
make
```

3.1.4 Hlavní vlastnosti

Mezi hlavní vlastnosti nástroje SIPp patří generování jednoho či více hovorů do vzdáleného systému.[11] Jednou z možností spouštění je příkazový řádek. Následující příklad ukazuje spuštění dvou instancí SIPp na jednom systému, k otestování jeho možností. Na přístroji, s nainstalovaným programem SIPp, je možné spustit předdefinovaný scénář UAS:

```
./sipp -sn uas
```

Na stejném přístroji je pak možné spustit klienta, který bude komunikovat právě se serverem. To je možné díky předdefinovanému scénáři UAC:

```
./sipp -sn uac 127.0.0.1
```

Tímto jednoduchým příkladem lze naprosto bez problémů po instalaci zjistit možnosti programu a také vyzkoušet možnosti ovládání programu. Je dobré připomenout, že spouštění nástroje je nutné ze složky, do které byl program nainstalován.

3.1.5 Integrované scénáře

V programu SIPp existuje několik předem definovaných scénářů, jenž je možné využít pro práci.[11] Tyto scénáře slouží hlavně k základnímu otestování funkcí tohoto programu. Je možné vytvářet i vlastní scénáře (kapitola 3.1.9). Jak již bylo zmíněno SIPp obsahuje několik předem definovaných scénářů, jedná se konkrétně o:

- UAC – Klasický klient.
- UAC s médii – Klasický klient odesílající média.
- UAS – Klasický server.
- Regexp – Scénář pro práci s regulárními výrazy.
- Branch – Jedná se konkrétně o dva scénáře pracující proti sobě, *branchc* a *branchs,* sloužící pro vyzkoušení práce s nelineárními cestami a větvením.
- UAC OOC – Pokud SIPp obdrží žádost *out-of-call,* tak je automaticky iniciován tento scénář, který jednoduše odpovídá *200 OK*.
- 3PCC – Jedná se o scénář pro práci s třetí stranou. Konkrétně existují 4 tyto scénáře: *3pcc-A, 3pcc-B, 3pcc-C-A, 3pcc-C-B*.

3.1.6 Ovládání programu SIPp

SIPp může být interaktivně ovládán skrze klávesnici či UDP soket. SIPp podporuje jak klávesové zkratky, které je možné zadat kdykoliv během chodu programu, tak jednoduchý příkazový režim. V příkazovém módu (do něhož je možné vstoupit po stisknutí klávesy „*c*“) je možné napsat příkaz, který instruuje SIPp k vykonání nějaké akce. Příkazový mód je mnohem univerzálnější, než ovládání pomocí kláves, nýbrž zadávání příkazů je složitější a mnohdy pomalejší způsob. Veškeré možnosti ovládání programu jsou popsány v přehledných tabulkách na stránkách programu.[11]

3.1.7 Ovládání provozu

Nástroj SIPp generuje SIP provoz v závislosti na použitém scénáři. Je tak možné ovládat počet spuštěných hovorů, které jsou započaty každou sekundu. Ovládání je umožněno pomocí dvou výše zmíněných metod (klávesou či z příkazového módu), stejně jako je ovládání provozu možné pomocí parametrů, zadávaných při startu programu. Všechny eventuality zadávání parametrů při spouštění programu jsou popsány na stránkách programu.[11]

3.1.8 Spouštění SIPp na pozadí

SIPp může být spuštěn jako program na pozadí. K tomuto účelu slouží jednoduchý příkaz „*-bg*“ zadávaný při spuštění programu v příkazové řádce. Tímto bude SIPp oddělen od stávajícího terminálu a bude pracovat na pozadí. Pokud nebyl zadán počet volání, které mají být vykonány, možností „*-m*“ při spouštění v příkazové řádce, tak bude SIPp běžet navždy. Existuje mechanismus, pomocí něhož je možné SIPp běžící na pozadí korektně ukončit. Příkaz `kill -SIGUSR1 [SIPp_PID]` předá instrukce programu SIPp běžícímu v pozadí, že má ukončit všechny probíhající hovory a zastavit všechny činnosti spojené s generováním nových hovorů.[11]

3.1.9 Tvorba vlastních XML scénářů

Jak již bylo výše popsáno, v programu SIPp je možné vytvářet vlastní XML scénáře i přesto, že jsou již od tvůrců před vytvořené základní scénáře. Ovšem tyto scénáře nevystihnou všechny možnosti, jež SIPp nabízí a právě proto, je možné vytvářet scénáře vlastní, které mohou prakticky jakoukoliv situaci popsat a později důvěrně nasimulovat. Ve scénářích je možné využít zpráv jako *INVITE, ACK, BYE,* stejně jako mnoho dalších. Scénáře lze pojmout jak z pohledu klienta, tak ze strany serveru, čímž lze vytvořit vzájemně komunikující dvojici simulující reálný systém, či je možné simulovat pouze jednu stranu spojení.[11]

3.1.10 Struktura tvorby scénáře

K vytvoření vlastního scénáře není mnoho zapotřebí. Tím, že scénáře mají jednoduchou strukturu, která se skládá jen ze dvou povinných částí (a dalších částí, které jsou čistě na uvážení

a potřebách uživatele), jsou pro uživatele poměrně lehké na vytvoření. Scénář musí vždy začínat takto:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<scenario name="Jméno scénáře">
```

Začátek každého scénáře není obtížný, kde první řádek znamená pouze ukazatel, že se jedná o XML soubor a také to, jaké kódování bude použito. Druhý řádek slouží k úvodu do scénáře. V uvozovkách je uvedeno jméno scénáře. Na konci každého scénáře pak musí být uveden pouze jeden řádek, který označuje ukončení scénáře a po něm se dále nebude v čtení XML souboru pokračovat. Konkrétně musí být napsáno následující:

```
</scenario>
```

Mezi druhým řádkem úvodu a posledním řádkem je pro uživatele naprostá volnost. Ten pak může využít spoustu atributů, díky kterým pak může vytvořit víceméně jakoukoliv situaci, která může u SIP protokolu nastat. Veškeré možnosti tvorby scénářů jsou vypsány na stránkách programu.[11]

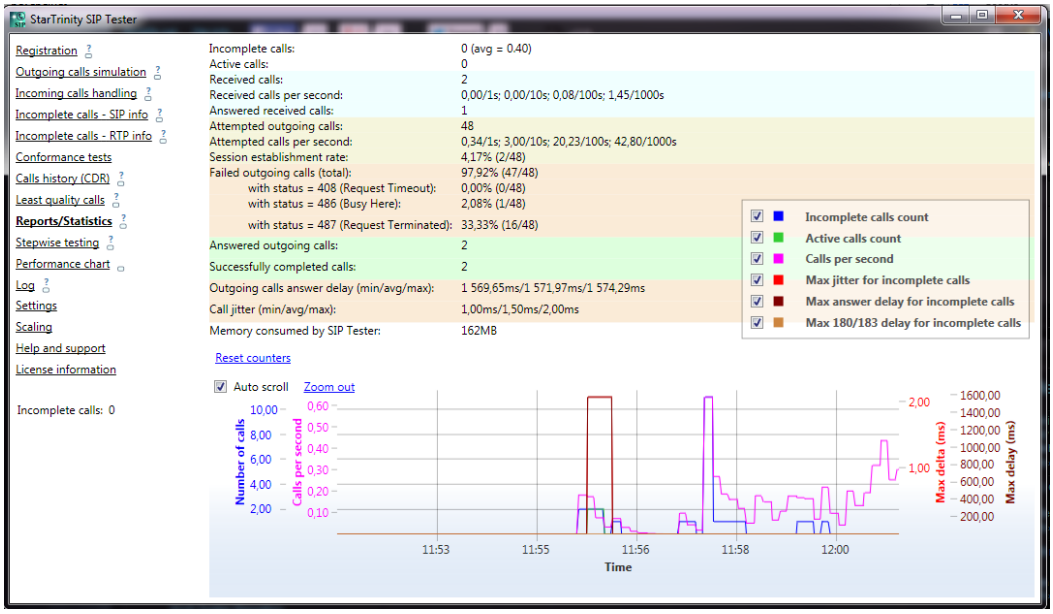
3.2 Star trinity

StarTrinity SIP tester™ (dále jen ST) je nástroj, který je určen pro sledování a testování VoIP sítí a SIP softwaru nebo hardwaru. ST umožňuje simulovat tisíce příchozích a odchozích hovorů s RTP, analyzovat kvalitu hovoru a vytvářet reporty v reálném čase. Běh hovoru je specifikován pomocí *CallXML,* kde je možné navrhnout různé situace z reálného systému. ST pracuje na jakémkoliv zařízení s Windows bez nutnosti speciálního hardwaru, simulačního aplikačního serveru, serveru pro média, SIP telefonu nebo registrátoru. Existují dvě verze tohoto programu. Jedna z nich je volně šiřitelná a druhá je komerční. Komerční je dostupná pouze na žádost u tvůrců. Volně šiřitelná verze má omezený počet simultánních volání. Stručný výpis hlavních vlastností programu:[12]

- Měření odchylky zpoždění, procentuální ztrátovost paketů a zpoždění odpovědí u RTP.
- Analýza a zprávy v reálném čase o jednotlivých voláních či o všech voláních.
- CDR (Call Detail Records, záznam hovoru na ústředně) zprávy – základní informace o hovoru, RTP statistiky, zaznamenaný celý hovor. Možnost exportu do CSV souboru.
- Zpráva o nejmenší kvalitě hovoru pro rychle a snadné zobrazení nejhoršího hovoru.
- Hromadné vytváření SIP hovorů – manuální a na časovač.
- Příjem hovorů s registrací a bez registrace na SIP testeru.
- Simulace ztrátovosti paketů na SIP a RTP protokolech.
- UAC a UAS registrace.
- SIP přes TCP a UDP.
- SIP autentizace.
- Podporované specifikace – RFC2833, RFC2976, RFC3261, RFC3262, RFC3264, RFC3362, RFC3515, RFC3550, RFC4028.

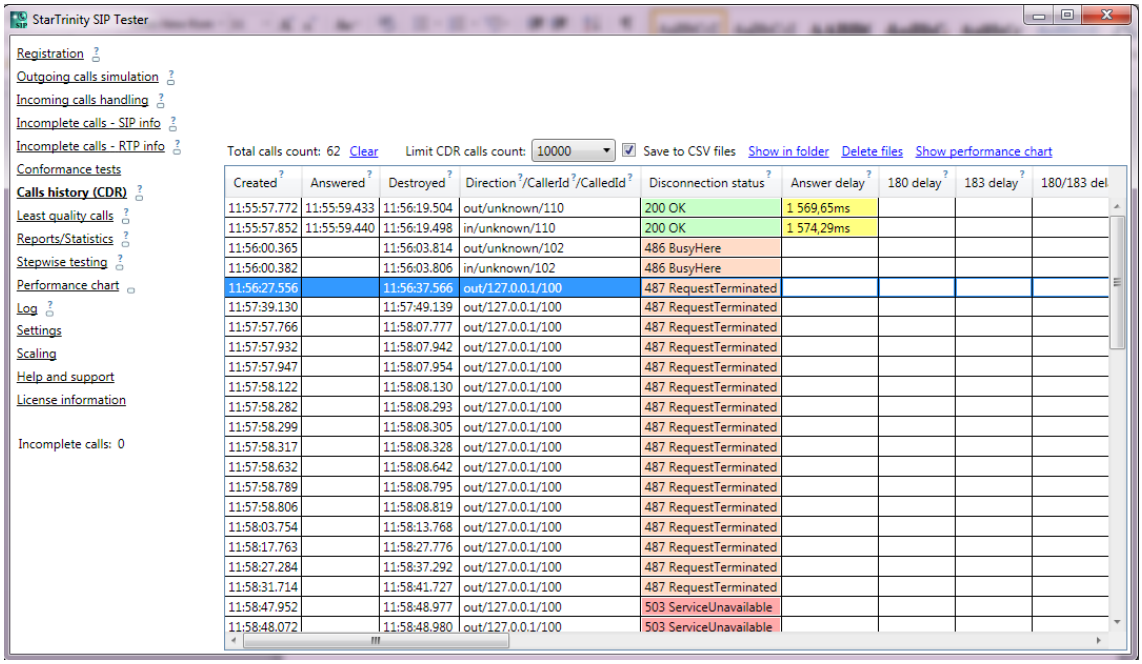
3.2.1 Instalace a práce s testerem ST

Instalace nástroje je poměrně jednoduchá, kdy je k dispozici klasický instalační soubor. Po spuštění programu se objeví úvodní obrazovka programu, po jejímž zobrazení je možné již s programem naplno pracovat a to konkrétně nastavením testů. Nastavování probíhá pomocí dvou dostupných možností. První z nich je pomocí přehledného GUI, kde uživatel zadá vše, co potřebuje. Druhá varianta je pomocí takzvaného *CallXML*. Po nastavení testu může začít samotná simulace hovorů. Simulace hovorů funguje poměrně dobře a vše zvýrazňuje přehledné grafické rozraní s výsledky testů. Na obrázku 3.1 je možné vidět graf, jenž je k dispozici po ukončení simulace hovorů.[12]



Obr. 3.1: *ST – graf simulace*

Uživatel má k dispozici i tabulku CDR (obrázek 3.2), kde je podrobně popsáno každé volání i s důvody selhání či naopak informace o úspěšnosti hovoru i s časy, kdy byl hovor odeslána a přijat.[12]



Obr. 3.2: *CDR tabulka*

Program v nekomerční verzi je bohužel omezen na 60 hovorů, což je pro testování výkonosti ústředny, stejně jako pro účely této diplomové práce, absolutní nedostatek.

3.3 PacketGen

PacketGen (dále jen PG) je paketový generátor pro generování hromadného VoIP provozu. PG generuje provoz jak se SIP signalizací, tak i s RTP pakety. Program dále slouží k zátěžovému testování a podrobné analýze VoIP sítě a umožňuje generovat až 2000 simultánních volání. PG je založen na distribuované architektuře se softwarovými jádry pro SIP a RTP, které jsou modulárně naskládány v jednom nebo více PC za účelem vytvoření škálovatelného velkokapacitního systému pro generování až 2000 testovacích volání. Nástroj PG může být použit pro testování funkcionality a ověření správné implementace protokolů v síti založené na SIP prvcích jako například: SIP telefon, síťový server, proxy server, registrátor, PSTN atd.[13]

3.3.1 Vlastnosti

Nástroj PG byl vytvořen pro plnění několika hlavních úkolů. Zkrácený soupis hlavních vlastností PG je sepsán v následném seznamu[13]:

- Kapacita
  - Distribuovaná architektura pro SIP a RTP systémy zajišťující vysokou míru volání a toky médií. PG může generovat až 2000 hovorů na dostatečně výkonném PC.
- Generování hovorů
  - Plná podpora SIP funkcionalit – registrace, přesměrování hovorů, přidržení hovorů, přenos hovorů, autentizace atd.

    - Manuální a hromadné možnosti volání s kompletní flexibilitou u každé relace volání.
    - Kompatibilní s RFC 3261, RFC 2833.
    - Generování jak SIP signalizace, tak RTP provozu.
    - Podpora změny parametrů za běhu ke kontrole volání a chování
- Řízení provozu
    - Možnosti vytváření skriptů pro generování RTP provozu za možností simulace a testu IVR.
    - Automatické generování poruch u RTP pro některé nebo všechny sestavené hovory.
    - Sledování testů IVR pro zjištění kvality dat a zvuku.
    - Automatické IVR testy – sestavení spojení a generování provozu řízené pomocí skriptů.
    - Zasílání či nahrávání zvukových souborů z některých nebo všech RTP relací. Možnost pozdější analýzy kvality zvuku.
- Podporované kodeky
    - PG podporuje téměř všechny průmyslově podporované kodeky.
    - EVRC-C není podporován.
- Vzdálený přístup
    - Vzdálený přístup je možný pomocí grafického uživatelského rozhraní, rozhraní příkazového řádku nebo vzdálené plochy.
- Zprávy
    - Zprávy poskytují záznamy hovorů, události a statistiky.

3.3.2 Použití

Nástroj PG má samozřejmě i velkou škálu využití. Je možné ho využít nejen pro klasické zátěžové testování (pomocí generování hovorů, jak manuálního, tak hromadného), ale také pro analýzu kvality hovoru, regresního testování, tzv. „Matrix“ testování či k testování protokolových vlastností a kompatibility kodeků.[13]

3.3.3 Instalace a práce s testerem PG

Instalace je možná po registraci na stránkách tvůrce softwaru. Kde jsou dále také popsány minimální požadavky na počítač, kde mezi základní požadavky patří počítač s operačním systémem Windows.[13]

3.3.3.1 SIP nastavení a konfigurace

Obrazovka nastavení SIP obsahuje všechny potřebné prvky pro určení parametrů relace. Uživatel má velkou flexibilitu při stanovování parametrů více SIP nebo RTP instancí na lokálním stroji, případně na vzdáleném stroji.[13]

3.3.3.2 Možnosti nastavení UA

Všem UA v systému je možné přiřadit hned několik parametrů, ty se konkrétně rozdělují do 4 kategorií:[13]

- Parametry SIP – Umožnuje nastavit uživateli hlavičky pro odchozí SIP/SDP zprávy. Uživatel dále může nastavit jméno UA, jméno hostitele, port, adresu SIP serveru, možnosti NAT a kontakty pro každé UA.
- Nastavení médií – Umožnuje uživateli nastavit RTP parametry pro UA. Ty slouží k určení možností schopností UA, co se týká přenosu médií. Tyto parametry se používají při vyjednávání charakteristiky hovoru během jeho sestavování.
- Další hlavičky – Možnost konfigurace dalších SIP/SDP hlaviček, které by měl každý UA použít. Tyto hlavičky budou zahrnuty ve zprávách zasílaných UA.
- UAC autentizace – Nastavení informací ohledně uživatelské autentizace nutné pro simulaci UAC.

3.3.3.3 Manuální a hromadné generování hovorů

Program PG podporuje jak manuální neboli jednotlivé generování volání, tak hromadné generování volání s možností nastavit jednotlivé hovory. Je možné rychle vytvořit několik UA, jež budou mít stejné nastavení jako již jeden vytvořený UA, čili se jedná o kopírování parametrů z jednoho UA na jiné. U každé relace lze vidět její aktuální stav či při generování provozu je opět proveditelné stanovování parametrů hromadné nebo manuální.[13]

3.3.3.4 SIP registrace

Nástroj PG umožňuje registrovat jeden nebo více UA během jednoho okamžiku. S tím, že každá registrace umožnuje různorodá nastavení jako adresu registrátoru, dobu platnosti,… Každá registrace může být také nakonfigurována tak, že se automaticky aktualizuje, pokud vyprší její doba platnosti. [13]

3.4 Asterisk Originate

Generování hovorů je možné také díky samotné ústředně Asterisk. Konkrétně se jedná o možnosti Originate, kdy se nabízí celkově tři možnosti využití.

3.4.1 Příkaz Originate

Příkaz ústředny Asterisk Originate slouží ke generování hovoru.[14] Poprvé se tento příkaz objevil ve verzi Asterisk 1.6.2.

- Syntaxe – *Originate(tech_data,type,arg1[,arg2[,arg3]])*
- Argumenty – Popis jednotlivých argumentů syntaxe:
  - *tech_data* – Technologie a datový kanál pro vytvoření odchozího hovoru např. SIP/1234.

- *type* – Typ, tento argument může nabývat hodnot „*app*“ nebo „*exten*“, v závislosti zda je odchozí kanál připojen k aplikaci či telefonní lince.
- *arg1* – Pokud je typ „*app*“, tak *arg1* udává jméno aplikace. Pokud je typ „*exten*“, tak *arg1* označuje kontext, jak bude kanál zaslán.
- *arg2* – Pokud je typ „*app*“, tak *arg2* znamená data poslané do aplikace. Pokud je typ „*exten*“, tak *arg2* označuje telefonní linku, kterou bude kanál zaslán.
- *arg3* – Pokud je typ „*exten*“, tak *arg3* znamená prioritu, se kterou bude kanál zaslán. Pokud je typ „*app*“, tak je *arg3* ignorován.

- Popis – Tato aplikace vytváří odchozí hovory a pojí je do aplikace, případně telefonní linky. Aplikace skončí, pokud odchozí hovor selže nebo dostane jakoukoliv odpověď. Jakmile aplikace skončí, tak si uloží hodnotu do proměnné a volací plán bude dále pokračovat.
  Aplikace uloží výstupní hodnotu do proměnné *${ORIGINATE_STATUS}*, která značí výsledek hovoru. Tato proměnná může mít několik možných hodnot:
  - *Failed*
  - *Succes*
  - *Busy*
  - *Congestion*
  - *Hangup*
  - *Ringing*
  - *Unknown*

3.4.2 Originate pomocí příkazového řádku

Pomocí příkazového řádku ústředny Asterisk je také možné vygenerovat hovor.[16] Jedná se o příkaz Originate, který může vygenerovat hovor a propojit ho s aplikací nebo kontextem.

- Popis – Existují dvě možnosti jak využít tento příkaz v příkazové řádce. Hovor může být sestaven mezi hovorovým kanálem a specifickou aplikací anebo mezi hovorovým kanálem a telefonní linkou ve volacím plánu.
  - První možnost - *channel originate <tech/data> application <jméno aplikace> [data aplikace]*
    Tato možnost vytvoří hovor mezi hovorovým kanálem a specifickou aplikací.
  - Druhá možnost - *channel originate <tech/data> extension [<exten>@][<kontext>]*
    Tato schopnost vytvoří hovor mezi hovorovým kanálem a specifickou telefonní linkou.
- Příklad – pokud je v *extensions.conf* uvedena tato konfigurace:

```
[greeting]
exten => 400,1,Answer
exten => 400,n,Background("hello")
exten => 400,n,Wait(5)
exten => 400,n,HangUp()
```

Tak poté příkaz v příkazové řádce musí vypadat následovně:

```
originate SIP/1/123456 extension 400@greeting
```

3.4.3 API Asterisk Originate

Příkaz ústředny Asterisk Originate je možné využít ještě jedním způsobem.[15] A to konkrétně jako jednu možnost Asterisk API.

- Parametry:
  - *Channel* – Kanál, na kterém bude hovor vygenerován.
  - *Context* – Jaký kontext bude použit (musí být nastaveno *Exten* a *Priority*).
  - *Exten* – Jaká telefonní linka bude použita (musí být nastaveno *Context* a *Priority*).
  - *Priority* – Jakou prioritu bude mít vygenerovaný hovor (musí být nastaveno *Context* a *Exten*).
  - *Timeout* – Časový limit v milisekundách pro vytvoření spojení.
  - *CallerID* – ID účastníka spojení.
  - *Variable* – Proměnné pro nastavení kanálu (maximálně 32).
  - *Account* – Účet pro volání.
  - *Application* – Aplikace použitá pro volání.
  - *Data* – Data pro aplikaci, pokud je parametr aplikace použit.
  - *ASync* – Parametr pro možné asynchronní volání, tedy není nutné čekat na odezvu před vytvořením dalšího volání.
  - *ActionID* – Sekvenční číslo žádosti.
- Sled událostí – prvním krokem je kanál. Poté, pokud je přijata odpověď, se volí telefonní linky s kontextem pro vytvoření druhého konce volání. Parametr *timeout* se vztahuje pouze k počáteční tvorbě kanálu. Použití ASync vede k tomu, že je uskutečněna událost *OriginateResponse* , která obsahuje zdroj chyby, pokud se nějaká chyba vyskytla. Důvody mohou býti následující:
  - 0 = špatné telefonní číslo či linka
  - 1 = žádná odpověď
  - 4 = zodpovězen
  - 5 = zaneprázdněn
  - 8 = přeplněn nebo nedostupný

4 Popis vlastních XML scénářů

Vzhledem k nedostatečnosti předdefinovaných scénářů nástroje SIPp byly vytvořeny vlastní scénáře pro účely testování. Některé z nich jsou zaměřeny na standard RFC 6076, jiné zase simulují situace v reálném provozu či se snaží napodobit RFC 6076, jen s jinou žádostí, která slouží pro pozdější srovnání. XML scénáře byly vytvořeny nejen pro klientskou stranu komunikace, ale také rovněž pro serverovou stranu komunikace. Simulovat je tedy možné nejen komunikaci mezi SIP ústřednou a klientem, ale rovněž také mezi klientem a serverem. Všechny klientské scénáře jsou doplněny o zasílání médií, tak aby byl reálný provoz opravdu věrně nasimulován a nebyla měřena pouze signalizace protokolu SIP. V klientských scénářích je rovněž zasílání žádosti *BYE*, což značí, že veškerá komunikace, pokud bude hovor úspěšný, bude ukončována klientem. Veškeré příkazy přesně tak, jak byly u jednotlivých scénářů použity, jsou přiloženy na CD v textovém dokumentu Příkazy.txt.

4.1 Soupis vytvořených scénářů

Pro účely testování byly použity scénáře s metodami *INVITE, REGISTER* a *OPTIONS.* Dva scénáře byly vytvořeny pomocí žádostí *INVITE* a *REGISTER* a jeden scénář byl vytvořen pomocí metody *OPTIONS.* Ovšem ke každému scénáři byl i vytvořen server, což zdvojnásobuje počet vytvořených scénářů.

4.2 Scénáře s žádostí INVITE

Žádost *INVITE* skýtá velké možnosti při vytváření. Pro účely testování byly vybrány dva případy. Jeden slouží k zabezpečení výpočtu délky trvání SRD (viz popis kapitola 2.2.3). Další slouží k otestování zátěže při nutnosti autentizace pomocí metody *INVITE*. Všechny scénáře budou dále popsány ve vlastních kapitolách.

4.2.1 Žádost INVITE bez autentizace

Scénář popisující žádost *INVITE* bez autentizace má simulovat podmínky při měření parametru SRD, dle RFC 6076. Tento scénář je kompletně vyobrazen na obrázku 2.3. Oproti obrázku nastala drobná změna v tom, že čas může býti měřen i k odpovědi 100, což je ovšem popsáno v přidružené kapitole.

4.2.2 Žádost INVITE s autentizací

Scénář, poskytující analogické porovnání k času registrace u metriky RRD. Hlavním úkolem je tedy zažádat o spojení pomocí *INVITE* metody a následně sestavit hovor. Scénář nejprve zasílá žádost *INVITE*, která je ovšem serverem odmítnuta s požadavkem o následnou autentizaci. V druhém požadavku *INVITE* je již obsažena autentizační část, kterou server schválí a může následně být sestaven přenos médií. Čas měření je u tohoto scénáře stanoven jako čas

mezi úvodní metodou *INVITE* a časem úspěšného přijetí klienta k hovoru, tedy událostí *200 OK*.

4.2.3 Servery pro žádosti INVITE

Pokud jsou vytvořeny klientské scénáře, tak je dobré k nim vytvořit i servery, se kterými budou moci klientské scénáře spolupracovat. Tyto servery byly napsány tak, aby přesně simulovaly chování Asterisku v ideální situaci, tedy bez chybových odpovědí. To znamená, že pokud by Asterisk odpověděl na žádost *INVITE* například pomocí odpovědi *200 OK,* tak server se musí zachovat stejně. Jakožto má Asterisk uložené všechny informace o uživatelích, tak naprosto stejná situace jde napodobit i pomocí SIPp serveru. SIPp server má uloženy veškeré potřebné informace v souboru csv, kde jsou uloženy informace o heslech každého uživatele, které jsou nutné k autentizaci.

4.3 Scénáře s žádostí REGISTER

Naprosto stejná situace, které bylo docíleno s žádostí *INVITE,* nastává také u žádosti *REGISTER*. Rovněž i u této žádosti byly zvoleny dva scénáře a to jeden pro svou totožnost s metrikou RRD standardu RFC 6076 a druhý pro zjištění parametrů a rychlosti odezvy u registrace bez autentizace. Každý scénář s metodou *REGISTER* má stejný průběh, nejdříve je provedena registrace u registračního serveru a následně je sestaveno spojení za pomocí metody *INVITE* a po navázání spojení přenášeny média.

4.3.1 Scénář REGISTER bez autentizace

Tento scénář, tedy žádost *REGISTER* bez autentizace, má sloužit k otestování možností metody *REGISTER* v případě, že není vyžadována autentizace. Tento scénář působí také jako analogie ke scénáři *INVITE* bez autentizace. Konkrétní čas měřený u této metody je čas registrace tzn. čas mezi událostmi *REGISTER* a *200 OK*.

4.3.2 Scénář REGISTER s autentizací

Tento scénář, provádějící žádost *REGISTER* s autentizací, je určen k měření metriky RRD popsané ve standardu RFC 6076. Situace týkající se tohoto scénáře i čas, jenž je měřen, jsou naprosto přesně vyobrazeny na obrázku 2.1. Jakmile dojde k úspěšné registraci, tak následuje zaslání metody *INVITE* a po přijetí do konverzace ze strany serveru jsou přenášeny média. U tohoto scénáře bude rovněž sledována metrika IRA (popsaná v kapitole 2.2.2). Počet neúspěšných registraci je samozřejmě také velice důležitý, neboť i tento údaj hodně napovídá o schopnostech daného registračního serveru.

4.3.3 Servery pro žádosti REGISTER

Vzhledem k vytvoření klientských scénářů s žádostí *REGISTER*, je nutné i pro scénáře se žádostí *REGISTER* vytvořit scénáře pro servery rovněž věrně simulující chování Asterisku.

Tyto servery se prakticky v ničem neliší od těch popisovaných v kapitole 4.2.3, kromě toho, že jsou určeny pro žádost *REGISTER*, tak tím pádem očekávají příjem žádostí o registraci, nikoliv žádosti *INVITE*. Ta samozřejmě také figuruje ve scénářích, nýbrž až po úspěšné registraci. Servery pro ověřování identity uživatele používají regulární výrazy, díky kterým mohou extrahovat jen požadované informace ze SIP žádosti zaslané klientem. Regulární výraz získává informace pouze o části hlavičky *Authorization* následujícím způsobem:

```
<ereg regexp="Digest .*username=\"([^\"]*)\"" search_in="hdr"
    header="Authorization:" assign_to="junk,username" />
```

Z hlavičky jsou tak získávány informace, nutné pro možnost ověření autentizace. Tyto získané údaje jsou pak porovnány s daty uloženými v csv souboru, který nahrazuje registrační server se všemi uloženými údaji.

4.4 Scénáře s žádostí OPTIONS

Metoda *OPTIONS* je poměrně odlišná od dvou popsaných v předcházejících kapitolách. Tato metoda dokáže zjistit informace ohledně možností volajícího i bez sestavení volání. Toto by ovšem způsobilo velké rozdíly mezi jednotlivými scénáři a proto je o po této metodě sestavena výměna informací mezi účastníky za pomocí metody *INVITE*.

4.4.1 Klientský scénář OPTIONS

Klientský scénář se žádostí *OPTIONS*, zasílá tuto žádost na server v očekávání odpovědi ze strany serveru. Čas je poté měřen mezi událostí, při které je odeslána žádost *OPTIONS* a časem odpovědi *200 OK*. Vzhledem k tomu, že čas u žádosti *OPTIONS* není rozhodujícím faktorem, tak je pro tento scénář měřen počet úspěšných odpovědí právě na žádost *OPTIONS*.

4.4.2 Serverový scénář OPTIONS

Jakmile je uveden scénář popisující klientskou stranu u žádosti *OPTIONS*, tak nesmí chybě jeho protistrana a to konkrétně server věnující se odpovědím na tuto žádost. Tento scénář opět pracuje na velice podobném principu jako předchozí popsané serverové scénáře. Jedinou změnou je opět očekávání jiné žádosti než u předem popsaných serverů.

5 Generátor volání

Generátor zátěžových testů je, se scénáři, základní součást celé práce. V této kapitole budou postupně popsány veškeré funkce generátoru, včetně použitých programů i návodu na to, jak s tímto generátorem pracovat. Generátor může pracovat se všemi druhy scénářů, to jak předdefinovanými, tak uživatelem vytvořenými. Výstup z generátoru je realizován do dvou csv souborů. Jeden soubor popisuje počet vytvořených hovorů, počet úspěšných a neúspěšných hovorů a mnoho dalšího. Druhý soubor se zajímá o počet jednotlivých SIP zpráv, které byly během testu zpracovány či odeslány. Dohromady tyto dva soubory poskytují dobrý přehled o tom, jak celý test proběhl, včetně započtených časů u metrik, které jsou pro tuto práci nejdůležitější a to konkrétně metrik standardu RFC 6076. V případě, že scénář neodpovídá žádné z metrik standardu RFC 6076, tak jsou měřeny časy nejblíže odpovídající nějaké z těchto metrik.

5.1 Použité nástroje

Pro zpracování generátoru bylo využito několik prostředků. Mezi tyto prostředky se řadí několikrát zmíněný SIPp, dále skriptovací jazyky Bash a php, značkovací jazyk html a v poslední řadě také knihovna JpGraph, která pomocí php umožňuje vykreslovat grafy.

Nejdříve bude stručně popsán samotný generátor SIPp (rozsáhlý popis se nachází v kapitole 3.1). Program SIPp byl použit ve velmi upravené verzi. Jedná se o verzi, která byla přizpůsobena tomu, aby odpovídala lepší podpoře pro přenos RTP paketů. Je to speciálně upravená verze komunitou uživatelů SIPp. Nejedná se jen o lepší verzi ohledně podpory RTP, ale také dalších věci, které se týkají stability programu a podobně.

Dále byl využit jazyk php. Tento jazyk je určen pro vytváření webových stránek a přesně této vlastnosti využívá i generátor. Přesněji je použita kombinace php spolu s html. Pomocí html a metody *POST* jsou data, zadávána uživatelem, zasílána jazyku php ke zpracování a odeslání dále do systému, lépe řečeno skriptům v jazyce bash, pro vykonání dalších potřebných akcí. Spolu s jazykem php byla využita i knihovna JpGraph, která umožňuje jednoduchou tvorbu grafu. Možnosti knihovny byly použity při vykreslování grafu týkajícího se počtu SIP zpráv a žádostí u posledního proběhnutého testu.

Poslední nástroj je již zmíněný skriptovací jazyk bash. Je to jazyk umožňující spouštění unixových příkazů a také mnoha dalších věcí, které jsou použity v tomto generátoru. Všechny skripty jsou spouštěny z php a vzhledem k tomu, že je nutné pro drtivou většinu příkazů v linuxových systémech mít root práva, tak i skripty jsou takto spouštěny.

5.2 PHP část

Následující kapitola se bude zaobírat popisem php části generátoru. Celkově se php část generátoru skládá z několika php souborů, pomocných souboru pro ukládání zadaných dat,

obrázků a css souboru se styly. Následně budou popsány všechny části i to jak celý generátor pracuje v php části.

5.2.1 Úvodní strana

Úvodní strana celé webové části generátoru je tvořena souborem index.php (přiložen na CD). Celá tato strana je optimalizována pro internetový prohlížeč Mozilla Firefox. Na úvodní straně může uživatel spatřit úvodní formulář, který umožňuje zadávání příkazu SIPp právě přes tento formulář. Na obrázku 5.1 je možné spatřit právě úvodní obrazovku celého generátoru.

Obr. 5.1: Úvodní obrazovka

S touto obrazovkou pracuje uživatel prakticky po celou dobu práce s generátorem, výjimku tvoří případné zobrazení statistik. Po spuštění webového rozhraní musí nejdříve uživatel zadat data, tak aby mohl být SIPp příkaz spuštěn. Ne všechna pole jsou povinná pro to, aby byla zadána. Ty řádky, které jsou označeny tučně, v části „Zadejte parametry příkazu“, je nutné vyplnit. Ostatní záleží na zvážení uživatele, zdali je chce vyplnit nebo ne. Vše je odvozeno od tvorby příkazu SIPp [11].

Dále se na úvodní straně nachází část „Nutné vyplnit“. V této části jsou čtyři položky. První z nich se vztahuje k tomu, kde v PC má uživatel uloženy skripty, které jsou nutné pro další běh programu (popis v kapitole 5.2.5). Druhá položka určuje složku, kde budou ukládány výsledky. Třetí položka označuje jméno souboru, do něhož budou výsledky ukládány. Jako poslední je čas, jak dlouho má být SIPp ukončován korektní cestou, než bude zastaven nekorektní cestou. V této položce tak uživatel volí čas korektního ukončování programu, což je velice důležitá položka. Například pokud hovor bude trvat tři sekundy (jako v případě této práce), tak je vhodné zvolit čas minimálně 3 sekundy. Vzhledem k tomu, že SIPp je zakončen, až jsou všechny hovory náležitě ukončeny, tak při trvání hovoru tři sekundy, je právě tento čas

minimální doba po níž SIPp bude nadále pracovat a čekat na zakončení všech probíhajících hovorů. Pokud by byla nastavena menší časová doba, tak násilné zakončení činnosti nástroje SIPp by mělo za následek nekompletně sesbíraná data. Ideální doba, pro trvání hovoru tři sekundy, na ukončení, je přímo úměrná zátěži, která je posílána směrem k VoIP ústředně.

Pod těmito položkami se nachází tlačítko „Potvrdit nastavení“, které uloží nastavení, přesně tak, jak to uživatel uvedl do jednotlivých polí a zobrazí uživateli přesný tvar zadávaného dotazu. Pokud je uživatel spokojen s tvarem dotazu, tak jak ho zadal, což si může zkontrolovat ve spodní části obrazovky, kde pod tlačítkem „Potvrdit nastavení“ je dotaz zobrazen. Tak není nic jednoduššího než dotaz, a tím i program SIPp, spustit tlačítkem „Spusť“. Pokud ovšem má uživatel výhrady k parametrům a rád by je změnil, tak je to možné učinit jednoduchým přepsáním dané položky a opětovným kliknutím na tlačítko „Potvrdit nastavení“. Dotaz je vždy zobrazen pro možnost snadné orientace v tom, co je zrovna nastaveno. Veškeré nastavení, tak jak je zadáno uživatelem, je ukládáno do pomocného souboru. Tento soubor zajišťuje možnost opětovného používání příkazu i po vypnutí webového rozhraní. V tomto souboru je tedy uložena vždy poslední použitá konfigurace.

Jakmile je program SIPp spuštěn, tak je možné jej zastavit jednoduchým klepnutím na tlačítko „Zastav“. Po stisknutí tlačítka je ukončen program SIPp, následně je vyčkáno několik vteřin pro korektní ukončení programu a zápis veškerých monitorovaných statistik do souboru. Pokud není SIPp do několika vteřin (čas si stanovuje sám uživatel) korektně ukončen, tak je využito ukončení běhu programu nekorektní cestou. Nekorektní zakončení by mělo nastávat pouze v případech špatného zadání času uživatelem či při nějaké chybě, která se mohla v programu vyskytnout a proto program nemohl být ukončen korektně. Následně jsou statistiky zpracovávány dalším skriptem, který vytvoří již konečné statistiky, jak je může vidět uživatel po skončení načítání webové stránky.

Na úvodní straně webového generátoru se rovněž nachází tlačítko „Obnovit“. Vedle tlačítka je napsáno, zdali je SIPp v provozu či nikoliv. Jednoduchým kliknutím na toto tlačítko se vždy obnoví stav, jestli je SIPp v provozu nebo ne. Poslední dvě tlačítka na úvodní straně jsou vztaženy k zobrazení statistik proběhlého testu. Bližší informace se nacházejí v kapitolách 5.2.4 a 5.2.5

5.2.2 Spouštění SIPp

Další část, týkající se php je tvořena souborem create.php (opět přiložen na CD). Tento soubor má konkrétně na starosti zavolání BASH skriptu, který spouští instance programu SIPp. BASH skriptu je předáván kompletní SIPp příkaz, který má být spuštěn. Tento příkaz získává php skript tak, že je mu zaslán pomocí metody *POST* z úvodní strany po stisknutí tlačítka „Spusť“.

5.2.3 Zastavení a zpracování výsledků

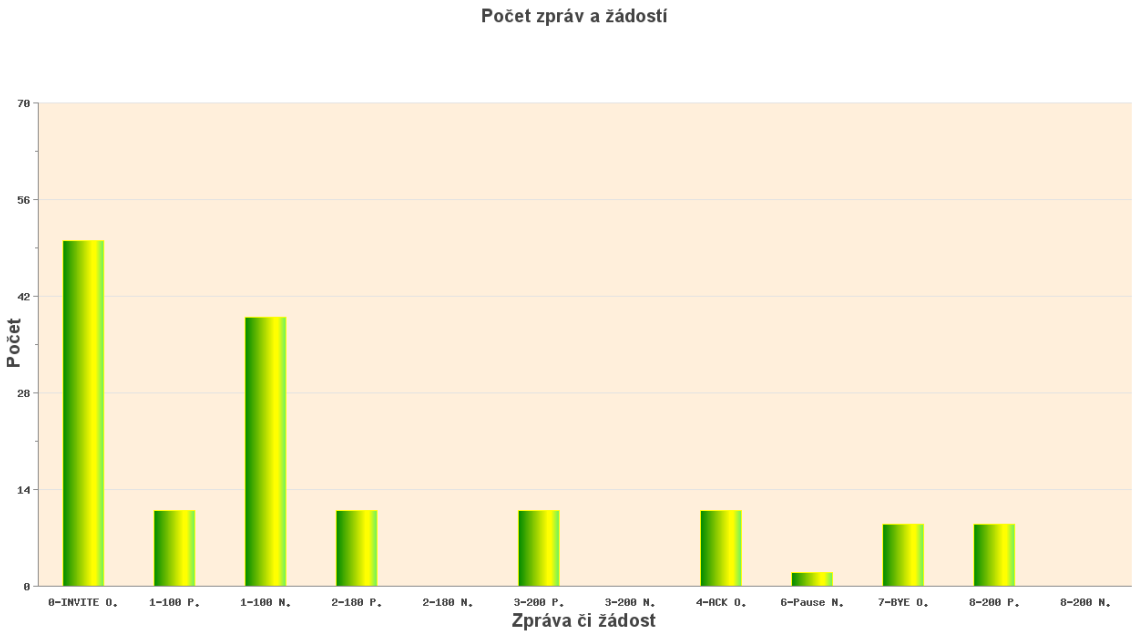
Následující skript obsažený v php části zastavuje spuštěnou SIPp instanci a následně převádí zaznamenané SIPp statistiky do finální podoby. To vše opět za použití BASH skriptů.

Rovněž jako v předchozím případě jsou data do tohoto php skriptu posílána pomocí metody *POST*.

5.2.4 Zobrazení grafů

Zobrazení grafu je další možností uživatele. Graf se konkrétně týká zobrazení počtu SIP žádostí a zpráv, které se vyskytly v daném testování. Kde a jaký soubor bude hledán, opět závisí na uživateli, neboť je prohledávána právě ta složka, která byla naposledy nastavena jako složka pro výsledky. To stejné platí i pro název souboru, jež je také odvozen od posledního zadaného jména v kolonce „Jak se má soubor jmenovat“. Veškeré informace zobrazené v grafu jsou uloženy i v souboru csv, ovšem graf poskytuje mnohem lepší vizualizaci, ve které lze jednodušeji vidět počty zpráv a tím i lépe rozeznat úspěšnost jednotlivých částí scénáře.

Graf je vykreslován za pomoci knihovny JpGraph. Tato knihovna je součástí složky www na přiloženém CD. Knihovna umožňuje vytvářet grafy jednoduchým způsobem s velkou možností nastavení, proto je ideální právě pro tvorbu grafu v této práce. V grafu je tak možné vidět veškeré příchozí i odchozí zprávy i žádosti. Ukázku grafu je možné vidět na obrázku 5.2.



Obr. 5.2: *Graf počtu SIP událostí*

5.2.5 Zobrazení statistik testu

K právě proběhlému testu je nutné zobrazit také statistiky. Tomuto účelu je určeno tlačítko „Informace ohledně testu“. Po kliknutí na tlačítko se uživateli zobrazí tabulka s informace o testu. Uživatel pomocí nastavení znovu rozhodne, o jakém testu chce mít informace zobrazeny. Tabulka je přehledně strukturovaná stejně jako csv soubor, tedy slouží k jednoduché orientaci pro uživatele. Příklad zobrazené tabulky je na obrázku 5.3.

Statistiky proběhlého testu

Počáteční čas	14.04.2014 13:23:11.536	14.04.2014 13:23:11.536	14.04.2014 13:23:11.536
Poslední resetovací čas	14.04.2014 13:23:11.536	14.04.2014 13:23:11.551	14.04.2014 13:23:18.358
Míra volání	10	10	10
Počet příchozích volání	0	0	0
Počet odchozích volání	0	50	50
Počet úspěšných volání	0	9	9
Počet neúspěšných volání	0	41	41
Čas odpovědi SRD(P)	00:00:00:000000	00:00:00:001000	00:00:00:000000
Čas odpovědi SRD(C)	00:00:00:000000	00:00:00:001000	00:00:00:001000
Délka volání	00:00:00:000000	00:00:00:610000	00:00:00:610000

Obr. 5.3: *Tabulka statistik testu*

5.3 BASH část

Samotné php skripty by samozřejmě nestačily vyplnit celou funkčnost generátoru. K tomuto účelu se perfektně hodí další skripty, tentokráte psané v jazyku BASH. Skripty řeší mimo jiné hlavní problém php a to nedostatek přístupových práv k zapisování souborů v operačních systémech založených na Unixu. Díky tomu, že Bash skript je možné spustit z php i jako root, tak je vhodné je právě propojit s php. Generátor využívá skripty k různým účelům. Všechny jsou popsány v dalších kapitolách.

5.3.1 Skript pro spuštění SIPp

Ačkoliv SIPp samotný nevyžaduje práva root pro spouštění, tak i tak je dobré spouštět instance programu s těmito právy. Obzvláště pokud jsou využívány scénáře, které zasílají média. Média jsou totiž uložena v souboru na disku a právě tak by mohl vzniknout problém. Skript, který spouští instance SIPp si jako parametr bere už hotový příkaz, který jednoduše spustí. Spolu se spuštěním instance programu je také spuštěno zaznamenávání statistik. SIPp má velké množství zaznamenávaných statistik a díky tomu, že statistiky zapisované tímto programem nespotřebovávají tolik výkonu, jsou vhodné pro využití i touto prací.

5.3.2 Skripty pro zastavení běhu

Podobná situace s přístupovými právy, jako u skriptu popsaném v předchozí kapitole, nastává také u skriptu pro zastavení běhu programu SIPp. Funkčně je skript svázán s tlačítkem „Zastav“. Jakmile uživatel zmáčkne tlačítko „Zastav“, tak jsou instrukce prováděny v následujícím pořadí. Pomocí php je nejdříve získáno identifikačního číslo procesu běžícího SIPp a s ním je pak dále nakládáno. Identifikační číslo procesu se konkrétně získává pomocí následujícího příkazu, jenž je poslán ke zpracování terminálu:

```
ps –e | grep sipp
```

Identifikační číslo procesu je následně využito při snaze o zastavení nástroje SIPp běžícího na pozadí následovně. Je zavolán první Bash skript, který se snaží zastavit běžící nástroj korektně, tak jak je to popsáno v kapitole 3.1.8. Jedině po stisknutí tlačítka „Zastav“ během běžícího programu je možné korektně zobrazit graf. Po uplynutí určitého času, který si musí uživatel sám stanovit na úvodní obrazovce, je znovu poslána žádost do systému o zjištění identifikačního čísla procesu běžícího programu SIPp. Dále jsou dvě možnosti. Jestliže žádné číslo nebylo obdrženo od systému, což značí, že byl SIPp korektně ukončen, tak se pokračuje v provádění dalšího php kódu. Ovšem zdali je od systému obdrženo identifikační číslo procesu značící předešlou neúspěšnou snahu o korektní ukončení procesu, tak je přistoupeno ke spuštění dalšího skriptu. Tento skript už má za úkol jedinou věc. Ukončit běžící SIPp za jakoukoliv cenu. K zastavení běžícího procesu je skriptem použit následující příkaz:

```
killall -9 sipp
```

5.3.3 Skript pro zpracování výsledků

Jakmile je ukončen běh programu, tak jako další v pořadí je spuštěn BASH skript, který veškeré SIPp statistiky zpracuje a vytáhne z nich vše pro uživatele důležité. Jako první věc, kterou skript vykoná, je přesunutí souborů se zaznamenanými statistikami tam, kde si zvolil uživatel. Další s tím spojená věc je přejmenování těchto souborů opět dle přání uživatele. Pro upřesnění, jsou přejmenovávány a přesouvány dva soubory. Jeden obsahuje informace například o časech odpovědí, či o celkovém počtu hovorů, jak úspěšných, tak neúspěšných atd. Druhý soubor obsahuje přesné informace o tom, kolik bylo posláno zpráv a žádostí během testu. Skript během jeho průběhu zapíše veškeré informace do uživatelem zvolených souborů, a jakmile ukončí svou činnost, tak je ukončena i práce php skriptu pro ukončení, který jinak do té doby prováděl načítání a tím dával uživateli najevo, že není vše řádně ukončeno. Následně si již uživatel může v jeho předem zvolené složce pro výsledky prohlédnout výsledný soubor.

5.4 Příprava na práci s generátorem

Pro práci s generátorem je třeba mít v systému velké množství knihoven a balíčků, které generátor využívá. Ovšem, ne jen instalace služeb, knihoven a balíčků, je zapotřebí ke správnému spuštění generátoru. Dále je také nutné provést velké množství změn týkajících se hlavně práv pro spouštění či zápis. Veškerá zde popsaná konfigurace je určená pro operační systém Ubuntu 12.04, na který je i celé webové rozhraní určené.

5.4.1 Instalace a konfigurace

Instalace začíná získáním balíčků nutných jak pro spuštění, tak pro překlad. Jména všech potřebných balíčků se nachází na CD v souboru Instalace.txt. Jakmile je instalace balíčků dokončena, tak je možné přistoupit k instalaci nástroje SIPp popsané v kapitole 3.1.3 Nicméně pro účely této diplomové práce a tohoto generátoru bude využita mírně pozměněná instalace. Na

přiloženém CD se nachází složka sipp. Jedná se o upravenou verzi nástroje SIPp, která má vylepšené vlastnosti týkající se přenosu a práce s RTP pakety a zároveň už jsou v této složce obsaženy i xml scénáře spolu s nutnými csv soubory. Celou tuto složku je nutné překopírovat do PC na vhodné místo na disku, odkud bude program následně spouštěn. Po instalací balíčků a SIPpu je instalační část u konce. Mírnou úpravou scénářů pak začíná konfigurace. V každém klientském scénáři je nutno upravit cestu k souboru s médii. Proto je nezbytné každý scénář otevřít pomocí textového editoru, například nano. Pro příklad otevření scénáře umístěného ve složce po překopírování.

```
nano /home/user/sipp-3.3/xml/invite_wo.xml
```

Po otevření souboru je potřeba najít ve scénářích sekci, kde je zapotřebí nahradit název složky, kde se nachází soubor. V případě popsaném níže by bylo změněno *student* na *user*.:

```
<nop hide="true">
<action>
<exec play_pcap_audio="/home/student/sipp-
3.3/pcap/g711a.pcap"/>
</action>
</nop>
```

Dalším krokem konfigurační části je zkopírování skriptů ze složky script z přiloženého CD kamkoliv do PC. Pro názornou ukázku bude předpokládáno, že skripty byly zkopírovány do složky `/home/user/sipp-3.3/script/`. Nyní je nutné přiřadit všem skriptů jejich práva. Za tímto účelem je nutné splnit dva následující kroky. Na začátek musí byt skripty spustitelné, tedy je přidělit skriptům práva pro spouštění. Docílení tohoto úkolu je možné následujícím příkazem:

```
chmod +x *
```

Samozřejmě je nutné nacházet se ve složce, kde jsou skripty uloženy, aby mohl přecházející příklad opravdu účinkovat. Druhým bodem při přiřazování práv skriptům je úprava práv root. Vše začíná otevřením souboru, ve kterém se práva specifikují.

```
visudo
```

Okamžitě po zadání předcházejícího příkladu se otevře soubor, ve kterém budou přiřazena práva každému skriptu. Proto je nutné dopsat na konec souboru následující řádky:

```
www-data ALL = (root) NOPASSWD: /home/user/sipp-
3.3/script/názevscriptu
```

Je nutné přiřadit přesně takový řádek pro každý skript, který je použit právě v této práci. Předcházející příkaz je pouze příklad, kde je uvedena složka jen pro názornou ukázku. Místo této složky je potřeba zadat přesný název složky, do kterého byly skripty zkopírovány.

Následně je možné zkopírovat obsah složky www z přiloženého CD do složky `/var/www/` a v tomto adresáři přiřadit práva pomocným souborů. Práva soubor budou přiděleny po vykonání příkazu:

```
chmod 666 hodnoty.txt
chmod 666 error.txt
```

Další neméně potřebnou věcí je přiřazení práv složce jpgraph, nacházející se ve složce `/var/www/`. Vše bude provedeno následujícím příkazem:

```
chmod 777 –R jpgraph/
```

Poslední konfigurační součást je umístění souborů se sadami znaků. Konkrétně obsah složky „fonty“ z přiloženého CD je nutné umístit do adresáře `/usr/share/fonts/truetype/`. Vše lze jednoduše provést příkazem *cp.*

```
cp /umístění/složky/fonty/z/CD/* /usr/share/fonts/truetype/
```

Tímto příkazem by měla být instalace a konfigurace generátoru u konce. Nezbývá než ověřit funkčnost. Spuštěním internetového prohlížeče, ideálně prohlížeče Mozilla Firefox, a zadáním adresy *localhost/index.php,* by se měla zobrazit úvodní strana webového rozhraní generátoru, tak jak je uvedena na obrázku 5.1.

5.5 Práce s generátorem

Jakmile je dokončena instalace všech potřebných prvků a je dokončena i konfigurace veškerých věcí nutných k bezproblémovému chodu webového rozhraní, tak je možné začít s tímto rozhraním a celým generátorem pracovat. Popis úvodní strany webového rozhraní generátoru je obsažen již kapitole 5.2.1. Nicméně v této kapitole nejsou všechny podrobnosti a možné funkce, stejně jako nutnost zadávání určitých parametrů v předem daném tvaru popsány. Proto se je tato kapitola pokusí shrnout.

Jak již bylo popsáno v kapitole 5.2.1, tak některé parametry je nutno zadat. Konkrétně je těchto parametrů 8, ale bez správného zadání nemůže generátor nikdy fungovat správně! Při zadávání parametrů týkajících se umístěná složky v systému (parametry „Umístění složky s programem“, „Kde jsou uloženy skripty:“, „Kde se mají ukládat výsledky:“) je nutné zadat složku v následujícím tvaru:

```
/home/user/sipp-3.3/
```

Jak je možné vidět v předcházejícím případě, tak tvar složky je nutné zadávat, jak s lomítkem na začátku, tak na konci. Stejné pravidlo ovšem neplatí při zadávání jména souborů pro ukládání výsledků, či při zadávání názvu scénáře. Tyto dva parametry jsou zadávány naopak bez lomítek na začátku a na konci. Scénáře, které jsou používány s generátorem, musí být uloženy ve složce, případně podložce, kde byl nainstalován program. U zadávání jména souboru pro výsledky je zadáván pouze název bez přípony.

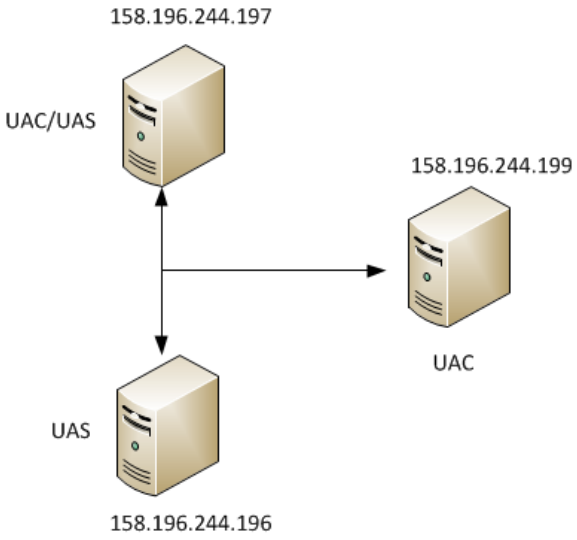
Ve spodní části generátoru je tlačítko „Graf“. Přesný popis se nachází v kapitole 5.2.4. Po stisku tlačítka je prohledávána složka a soubor podle posledního uloženého nastavení parametrů „Kde se mají ukládat výsledky:“, „Jak se má soubor jmenovat:“.

5.6 Testování

Další kapitola bude sloužit čistě k popisu metodologie testování. Bude vyobrazena testovací topologie včetně popisu hardwaru jednotlivých použitých strojů včetně operačních systémů. Dále bude naznačena konfigurace Asterisku i konfigurace jednotlivých strojů. Jako poslední bude nastíněno, jak probíhalo samotné testování.

5.6.1 Topologie a testovací hardware

Před samotným započetím testů je nutné stroje propojit do určité topologie, aby mohly navzájem komunikovat a tak si vyměňovat potřebné informace. Pro testování byly využity tři stroje, které byly zapojeny v následující topologii:



Obr. 5.4: *Topologie*

Na uvedené topologii je možné si všimnout, že každý stroj vystupoval v jiné roli. Stroj s IP 158.196.244.199 (dále jen Stroj 1) v topologii slouží jen jako klient. Zatímco stroj s IP 158.196.244.196 (dále jen Stroj 2) je v topologii čistě jako zástupce serveru, který bude testován. Přičemž stroj s IP 158.196.244.197 (dále jen Stroj 3) je využit jako klient, tak i server. Vše se odvíjí v závislosti na tom, jaký test je prováděn. Pokud je testován Stroj 2, tak je Stroj 3 využit jako klient generující zátěž. Ovšem pokud je Stroj 3 využit jako server, tak se topologie skládá jen ze dvou strojů a to konkrétně právě Stroje 3 jako serveru a Stoje 1 jako klienta generujícího zátěž. Jak bylo naznačeno, topologie je upravována v závislosti na tom, jaký stroj je testován. Při testu Stroje 2 je topologie shodná s obrázkem 5.4, přičemž zátěž je generována ze dvou zbylých strojů. Pokud je ovšem testován Stroj 3, tak Stroj 2 není využit a jediný stroj generující zátěž je Stroj 1.

U jednotlivých strojů je také nutné uvést specifikace hardwaru. Stroj 1 a Stroj 2 mají stejnou specifikaci vypsanou v následujícím seznamu:

- 8x Intel® Xeon® Processor E5-2660 @ 2.20 GHz
- 16 GB RAM
- OS Ubuntu 12.04 LTS

Stroj 3 má oproti dvěma předchozím odlišnou konfiguraci v podobě následujícího seznamu:

- Intel® Xeon® Processor E5-2660 @ 2.20 GHz
- 1GB RAM
- OS Ubuntu 12.04 LTS

5.6.2 Konfigurace strojů a Asterisku

Pro účely testování je nezbytné testované stroje připravit. Jednou z nejdůležitějších věcí je dostatečné nastavení maximálního počtu otevřených souborů. K tomuto úkonu je potřeba provést několik věcí. První z nich je nastavení limitů pomocí dvou následujících příkazů:

```
ulimit -n 262144
ulimit -s unlimited
```

Upravení limitů pokračuje upravením následujícího souboru `/etc/pam.d/login`, který je nutno otevřít v jakémkoliv textovém editoru, například nano, a to tak, že je jen přidáno nano před název souboru. Po otevření souboru je nutné přidat následující řádek:

```
session required /lib/security/pam_limits.so
```

Dále je vhodné upravit limity v souboru `/etc/security/limits.conf` opět jakýmkoliv textovým editorem, například editorem nano popsaným dříve. Po otevření souboru je nutné přidat na konec souboru následující dva řádky:

```
* soft nofile 500000
* hard nofile 500000
```

Konfigurace pak pokračuje posledním příkazem:

```
echo 262144> /proc/sys/fs/file-max
```

Limity jsou stanoveny poměrně vysoko, ovšem takto nastavené limity odpovídají stanovené zátěži a jakékoliv nižší hodnoty jsou riskantní. Po nastavení maximálního počtu otevřených souborů, je nutno dále přistoupit k instalaci a nastavení Asterisku. Vzhledem k tomu, že obsahem práce není instalace a konfigurace Asterisku (toto téma je tak rozsáhlé, že to jednoduše není v možnostech práce), tak je v práci zmíněn jen nejnutnější základ konfigurace Asterisku. Pro testování byly vybrány 4 verze Asterisku. Konkrétní je následující seznam:

- Asterisk 1.6.0.1
- Asterisk 1.8.15

- Asterisk 11.8.1
- Asterisk 12.1.1

Po instalaci jakékoliv verze je nutné vykonat několik kroků. Nejdříve bude muset být upravena hodnota v souboru /etc/asterisk/asterisk.conf. Jakmile je soubor otevřen, opět lze použít nano, tak je potřeba vyhledat řádek:

```
;maxfiles = 1000
```

Přičemž první krok je odstranění středníku za začátku řádku a následně je potřeba hodnotu 1000 přepsat na 262144. Řádek bude po změnách vypadat takto:

```
maxfiles = 262144
```

Dále je nezbytné přepsat hodnotu možných portů pro rtp spojení. Pomocí textového editoru musí být nyní otevřen soubor /etc/asterisk/rtp.conf. V tomto souboru se nachází řádek nutný k přepsání. Jedná se konkrétně o následující řádek:

```
rtpend=20000
```

Hodnotu 20000 v tomto řádku je pro bezchybný chod nutné přepsat na pohodlných 80000, což je sice velké číslo, ale zaručí tak hladký průchod i nejsložitějších testů. Třetím krokem pro bezproblémový chod je spuštění Asterisku jinak, nežli jako démona. Toto je provedeno následujícím příkazem, přičemž podmínka je zastavený Asterisk.

```
/usr/sbin/asterisk –c
```

Nyní je možné přistoupit k samotné konfiguraci vytáčecího plánu a jednotlivých účastníků. Nejdříve bude popsáno vytvoření účastníků. Do souboru *sip.conf* jsou postupně vloženi jednotliví účastníci, kteří později hrají roli při vytváření hovorů. Konkrétně se jedná o tři typy účastníků pro testování jednotlivých scénářů. Pro scénáře nevyžadující registrace, *INVITE* bez autentizace, *OPTIONS* a *REGISTER* bez autentizace je využita následující konfigurace účastníka.

```
[woauth]
type=friend
context=sipp
host=dynamic
username=woauth
canreinvite=yes
disallow=all
allow=alaw
User-ID="wauth"
```

Změnu vyžaduje uživatel, který je využíván pro scénář *INVITE* s autentizací. Vzhledem k tomu, že k autentizaci je potřebné heslo, tak je právě uživateli přiděleno. Opět je uvedeno přesné nastavení účastníka:

```
[wuser]
```

```
type=friend
context=sipp
host=dynamic
username=wuser
secret=wuser
canreinvite=yes
disallow=all
allow=alaw
User-ID="wuser"
```

A jako poslední popisovaný je účastník pro scénář *REGISTER* s autentizací. U tohoto účastníka je nezbytné změnit *type* na *peer* u verze Asterisku 1.6.0.1.

```
[wrauth]
type=friend
context=sipp
host=dynamic
username=wrauth
secret=wrauth
canreinvite=yes
disallow=all
allow=alaw
insecure=invite
User-ID="wrauth"
```

Těchto účastníku je možné vytvořit libovolný počet, samozřejmě s jinými jmény, ovšem tato práce využívala pěti účastníků z každé skupiny. V nastavení *general* byl uveden parametr `bindport=5062`, což zaručovalo komunikaci mezi UAC a UAS na portu 5062.

Pro chod Asterisku i celého generátoru musí být vytvořen vytáčecí plán. Pro účely práce byl vytvořen jednoduchý vytáčecí plán, kdy nejdříve začne Asterisk vyzvánět, později odpoví, poté je použita možnost Echo, kdy se zajištěn přenos médií oběma směry a nakonec Asterisk zavěsí, k čemuž by nemělo dojít, protože jako první by měl dle scénářů ukončit hovor klient. Konkrétní nastavení jedné linky je uvedeno na příkladu:

```
exten => 2005,1,Ringing
exten => 2005,n,Answer
exten => 2005,n,Echo
exten => 2005,n,Hangup
```

Kompletní soubory *sip.conf* a *extensions.conf* je potřebné přesunout soubory ze složky asterisk na přiloženém CD do složky `/etc/asterisk/`. Přesun obou souborů je samozřejmě nutné provést až je dokončena instalace Asterisku.

5.6.3 Praktické testování

Jakmile je veškerá konfigurace hotova, tak přichází na řadu testování praktické. Testovány byly všechny vytvořené scénáře stejným způsobem. Po nastavení parametrů spouštěného příkazu byl každý scénář puštěn dvě minuty. Tento čas by měl být dostatečný ke stanovení měřeného času, případně neúspěšných pokusů o registraci. Testovány byly všechny čtyři již zmíněné verze Asterisku ve stejné konfiguraci. Oba stroje byly rovněž testovány ve stejném stylu s jedním drobným rozdílem. Zátěž na slabší stroj, tedy Stroj 3 byla generována pouze ze stroje 1 a zátěž na Stroj 2 byla generovaná ze dvou zbývajících strojů.

Nyní už zbývá pouze uvést parametry, které byly při testování nastaveny. Všechno zmíněné bude v souladu s webovým rozhraním generátoru. U některých parametrů budou uváděny testované složky, či názvy souborů. Uživatel si samozřejmě může zvolit většinu z těchto parametrů dle svého uvážení.

- Umístění složky s programem - /home/user/script-3.3/
- Vlastní scénář
- Název scénáře: xml/invite_wo.xml
- IP adresa testovaného serveru: 158.196.244.196
- Port: 5062
- Služba: 2005
- Nastavení lokální IP: 158.196.244.199
- Umístění složky se souborem pro vkládání dat: xml/users_invite_wo.csv
- Počet hovorů za určitý čas: 50
- Maximální doba spojení: 120
- Rozlišení časovače: 50
- Limit nastavení watchdog časovače: 2000
- Nastavení přenosového módu: un

Tyto hodnoty jsou pouze ukázkové, ovšem velká část z nich zůstala po celou dobu stejná. Měněny byly pouze hodnoty „Název scénáře“, „IP adresa testovaného serveru“, „Nastavení lokální IP“, „Umístění složky se souborem pro vkládání dat“ a „Počet hovorů za určitý čas“. S tím, že „Název scénáře“ a „Umístění složky se souborem pro vkládání dat“ se měnil podle testovaného scénáře“, „IP adresa testovaného serveru“ dle IP adresy UAS, „Nastavení lokální IP“ podle IP adresy UAC a „Počet hovorů za určitý čas“ dle nutnosti.

Při jednotlivých testech bylo rovněž sledováno využití procesoru a paměti programem *htop,* který přehledně zobrazuje momentální využití procesoru a paměti. Pro testování byl vybrán kodek G.711a. Jedná se o evropskou verzi jednoho z nejpoužívanějších kodeků, proto je tato volba jedna z nejideálnějších. Tento kodek pracuje na vzorkovací frekvenci 8 kHz s rozlišením 8 bitů, to znamená přenosová rychlost 64 kbit/s. Při kódování signálu je použita logaritmická komprese, která je schopna převést dvanácti či třinácti bitový signál na osmibitový. V Evropě je pro tuto kompresi používán vzorec A-law, proto název G.711a.[19]

5.7 Možnosti s programem tcpdump

Možnosti zaznamenávání dat samozřejmě nejsou omezeny jen na statistiky měřené nástrojem SIPp. Nýbrž je možný výběr z několika druhů paketových analyzátorů. Jedním z těchto analyzátorů je i program tcpdump. Zaznamenávání hovorů pomocí nástroje tcpdump je v mnoha ohledech mnohem výhodnější než jen klasické zaznamenávání SIPp statistik. Díky programu tcpdump je možné mít detailní informace ohledně každého hovoru, který proběhl, včetně takových věcí, jako je ztrátovost jednotlivých hovorů atd.

Proto v části práce byla otestována i možnost zaznamenávání dat pomocí analyzátoru tcpdump a následné získání dat z výpisu tohoto programu. Při testování vlastností této možnosti zaznamenávání a analyzování dat se postupovalo následovně. Spolu se SIPp byl spuštěn i tcpdump, který zaznamenával data jen na určitém rozhraní a na určitém portu, tak aby zbytečně nezískával data nesouvisející s testováním ústředny. Jakmile byl ukončen provoz nástroje SIPp, tak ve stejný okamžik ukončil svůj provoz i tcpdump. Následně začalo získávání dat z výpisu analyzátoru tcpdump. Pro zjednodušení analyzátor zapisoval pouze hlavičky jednotlivých SIP zpráv a žádostí.

V okamžiku ukončení provozu analyzátoru začal pracovat skript, který vytahoval důležité informace o každé SIP zprávě a žádosti. Skript tedy získal informace ohledně času, kdy byla SIP metoda odeslána či přijata, z které IP pocházela a jaké IP byla určena. Dále získával informace o názvu metody, identifikačním čísle hovoru a také *branch* identifikující hovor.

Po implementaci funkce do webového rozhraní měl uživatel možnost vybrat si SIP metodu začátku a SIP metodu konce měření času. Tento úkon prováděl další skript, který počítal čas mezi dvěma metodami stejného hovoru. Tak bylo jednoduše možné zjistit, jak dlouho trvala doba právě mezi dvěma metodami neboli událostmi v jednom hovoru. Implementována byla i funkce, kdy si uživatel mohl vybrat jednu z metrik standardu RFC 6076 výběrem z rolovacího seznamu. Bylo tak možné zjistit přesný čas mezi zvolenými událostmi u každého hovoru.

Nicméně tento postup má dvě nevýhody. Jedna je ta, že analyzátor je další zátěž na již tak vytížený procesor. Tato nevýhoda by se dala jednoduše obejít připojením přepínače a analyzováním dat na přepínači. Ovšem druhá nevýhoda tohoto postupu je mnohem závažnější. Při zaznamenávání dat analyzátorem dochází k zaznamenávání velkého množství dat. Například u scénáře s žádostí *INVITE* s registrací je přibližně (dle odpovědí serveru) 10 SIP metod. Což značí asi 10 záznamů jen pro jeden hovor. Už při počtu volání deset za sekundu je to 100 záznamů za sekundu. Z tohoto obrovského kvanta dat pak skript získával jen potřebná data. Získávání dat tak zabralo velkou porci času. Konkrétně se jednalo o zpracovávání 300 hovorů asi 30 sekund a 1000 hovorů asi 180 sekund. K tomu je nutný připočíst čas trvání dalšího skriptu, který získával časy mezi dvěma metodami a z toho jednoznačně vyplývá, že takto postavený systém by v žádném případě nemohl v reálném systému pracovat.

6 Zhodnocení výsledků

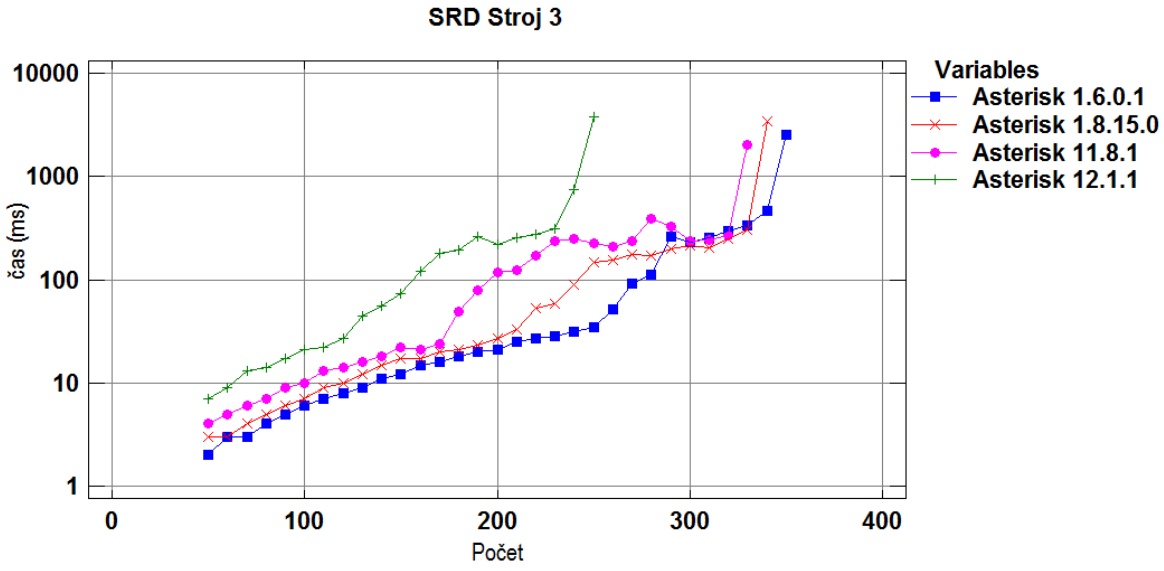
Po provedení praktické části, kdy byly všechny scénáře testovány oproti B2BUA, je zapotřebí vyhodnotit výsledky. Testy jako takové byly zaměřeny hlavně na časy odpovědí ústředny a samozřejmě také i na jiné prvky ovlivňující odpovědi. U některých scénářů byla zjišťována úspěšnost odpovědí a u téměř všech scénářů byl také ověřován vliv pamětí a procesoru na časy odpovědí. Na obrázcích a v grafech bude uvedeno počet volání za sekundu, což je při délce trvání jednoho hovoru tři sekundy až trojnásobný počet hovorů, které musí ústředna v jeden okamžik zvládnout. Například při uvedeném počtu 300 hovorů za sekundu znamená, že ústředna musí v jeden okamžik zvládnout až 900 hovorů. U všech hodnocených měření jsou uvedeny pouze grafy s tím, že veškeré tabulky jsou obsaženy v příloze a na CD. U všech grafů se na ose x nachází počet volání za sekundu (označen jen jako Počet).

6.1 Časy odpovědí

Jako první atribut bude vyhodnocen ten nejdůležitější měřený a to konkrétně čas. U každého scénáře, kromě scénáře se žádostí *OPTIONS* byl zaznamenáván čas mezi dvěma událostmi přesně tak, jak to popisuje RFC 6076. Případně u scénářů, které tomuto standardu nepodléhají, byl čas měřen co nejpodobněji standardu. Všechny uvedené grafy mají na časové (svislé) ose uvedené logaritmické měřítko pro lepší přehlednost.

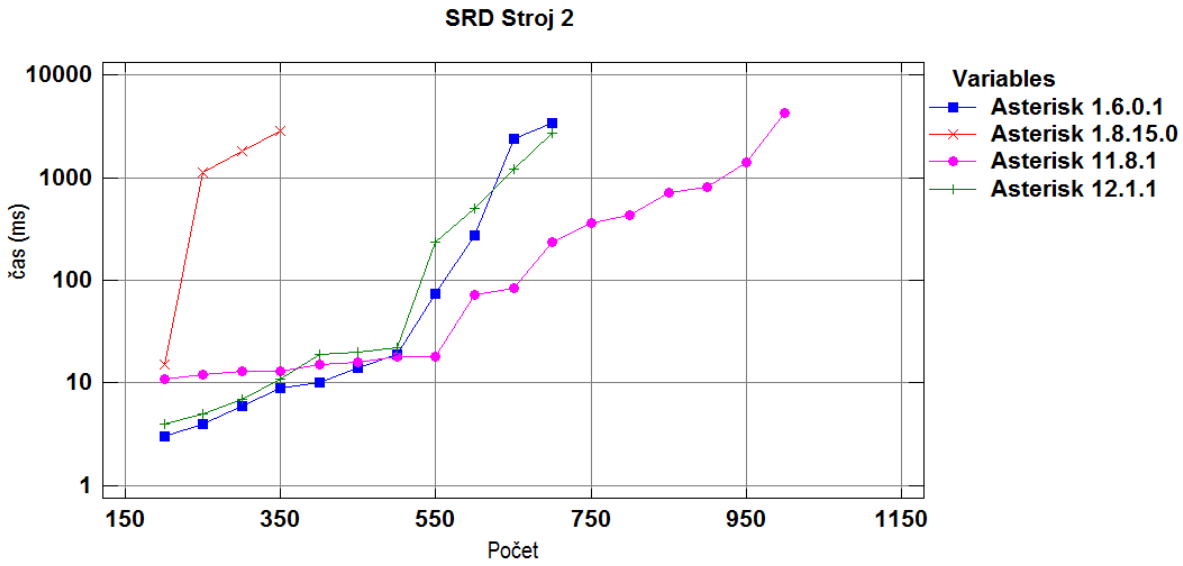
6.1.1 Čas SRD

Čas SRD je jedním ze základních pro určování výkonnosti SIP infrastruktury. Jako jeden ze dvou měřených časů vychází ze standardu RFC 6076. Jako všechny následující metriky a časy i tento byl měřen na dvou strojích, jejichž specifikace je popsána v kapitole 5.6.1.



Obr. 6.1: *Porovnání SRD u Stroje 3*

Jak je možné vidět z obrázku 6.1, tak u slabšího stroje není rozdíl mezi jednotlivými verzemi Asterisk až tak patrný. Jediná verze, která nezvládá více jak 300 volání za sekundu, je Asterisk 12.1.1.

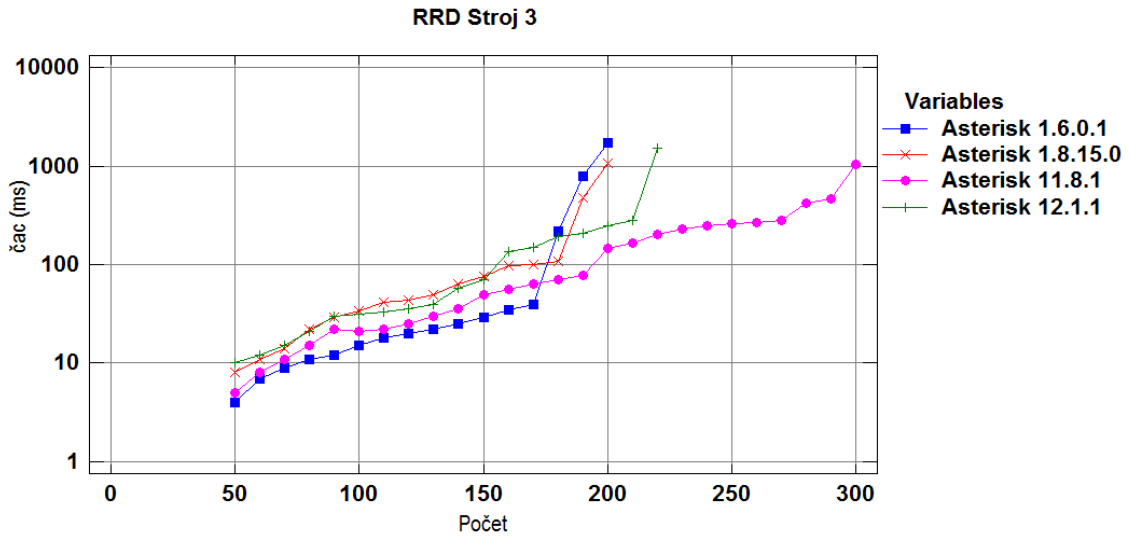


Obr. 6.2: *Porovnání SRD u Stroje 2*

U stroje 2 jsou rozdíly ve výkonosti jednotlivých verzí vidět zřetelně na první pohled. Nejlépe u testu dopadl Asterisk ve verzi 11.8.1, kdy zvládl skoro až tisíc hovorů za sekundu. Naopak Asterisk 1.8.15 nezvládnul efektivně ani počet hovorů blížících se 300 za sekundu.

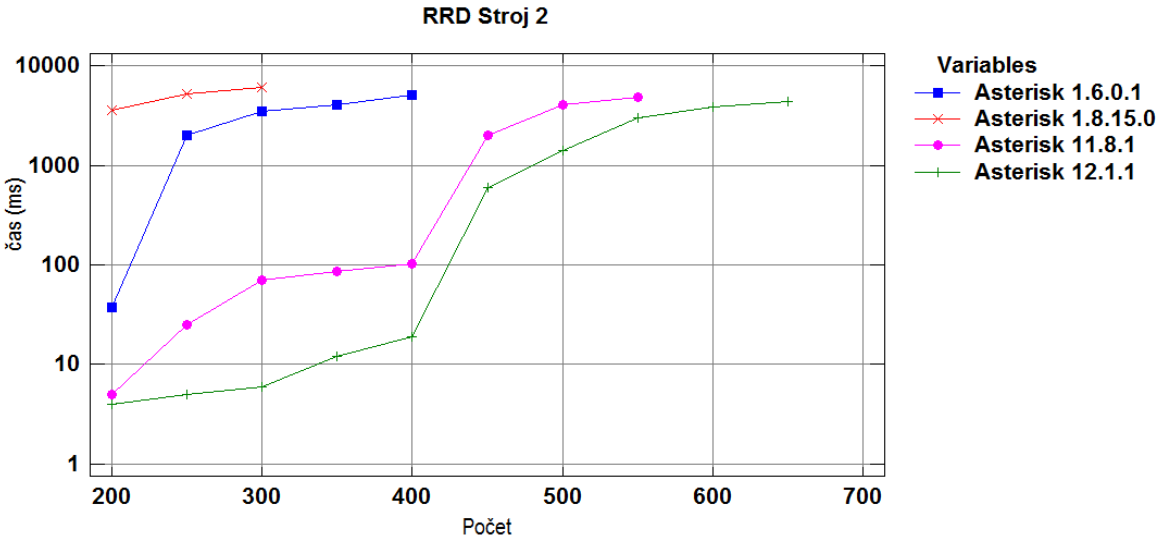
6.2 Časy RRD

RRD je další z metrik standardu RFC 6076, čili časy RRD jsou opět velice vypovídající statistikou ohledně schopností a parametrů ústředny. I u této metriky bylo zvoleno logaritmické měřítko v grafu a měření probíhalo na dvou strojích.



Obr. 6.3: *Porovnání RRD u Stroje 3*

Při pohledu na obrázek 6.3 je možné vidět patrný rozdíl oproti časům SRD. Žádost o registraci způsobil velké problémy hlavně dvou nejstarším verzím Asterisku. Z testu jednoznačně nejlépe vyšel Asterisk 11.8.1 a naopak nejhůře právě dva zmíněné Asterisk 1.6.0.1 a 1.8.15.0, kteří již při počtu volání 200 za sekundu měli problém s časy odpovědí.

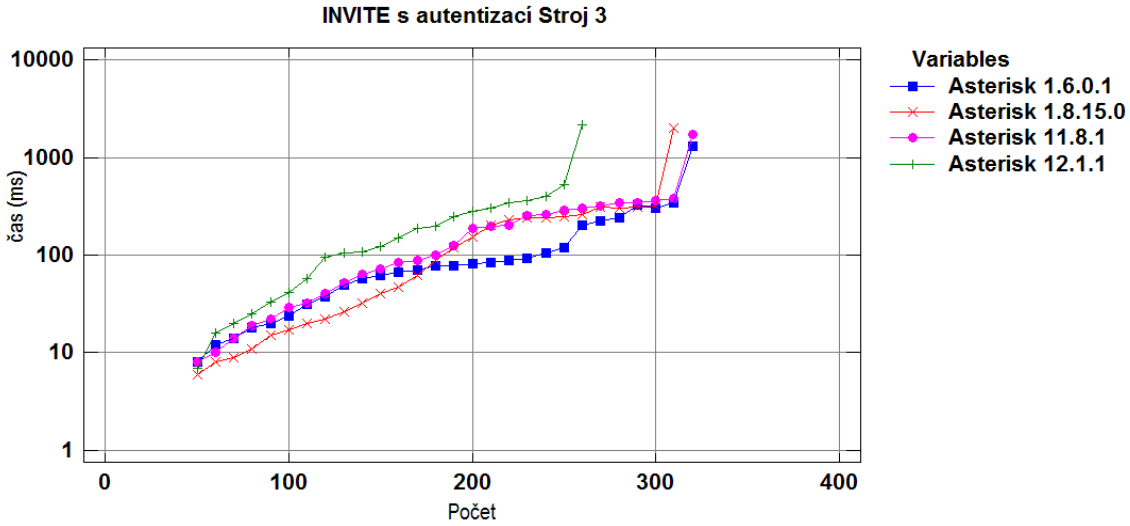


Obr. 6.4: *Porovnání RRD u Stroje 2*

Z přechozího grafu naopak jako absolutně nejlepší vychází Asterisk 12.1.1, který má nejlepší časy i při nevyšší míře hovorů za sekundu. Naopak znovu dva nejstarší typy Asterisku zaostávají už při velmi nízkém počtu volání za sekundu.

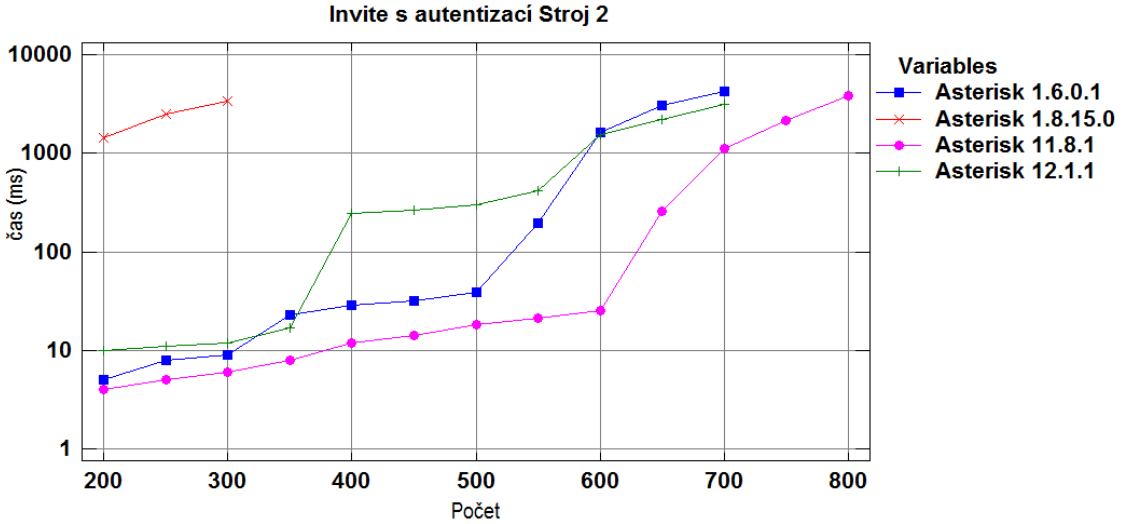
6.3 Časy INVITE s autentizací

Scénář žádosti *INVITE* s autentizací sice nepatří mez metriky RFC 6076, ovšem je velice vhodné, aby byl čas u tohoto scénáře měřen také. Čas, který je měřen je tak odvozen od času RRD a konkrétně je u tohoto scénáře měřen mezi první žádostí *INVITE* v dialogu a zprávou *200 OK*, která značí úspěšné přijetí žádosti.



Obr. 6.5: *Porovnání INVITE s autentizací u Stroje 3*

Popisovaný scénář neovlivňuje výsledky a rozdíly mezi jednotlivými verzemi Asterisku prakticky vůbec. Nepatrný rozdíl je u verze 12.1.1 kdy začínají časy vzrůstat o něco dříve, nicméně rozdíl není nijak markantní.

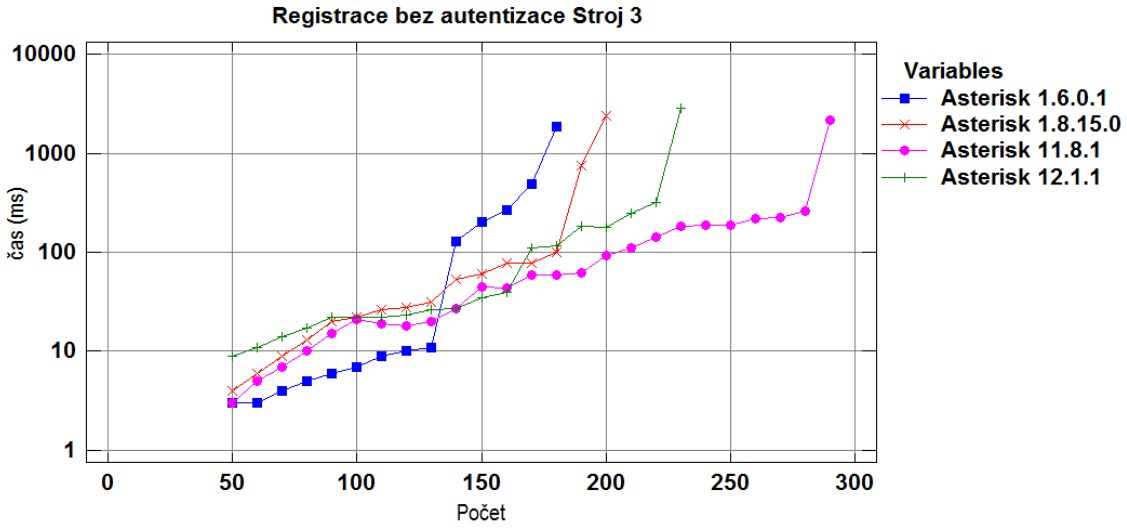


Obr. 6.6: *Porovnání INVITE s autentizací u Stroje 2*

Opět lze vidět, že Asterisk 1.8.15.0 výrazně zaostává při porovnání s jinými verzemi. Verze 12.1.1 a 1.6.0.1 jsou oproti tomu na podobné úrovni, kdy přestože má Asterisk 12.1.1 vyšší časy odpovědí u menších čísel, tak nakonec se situace obrací.

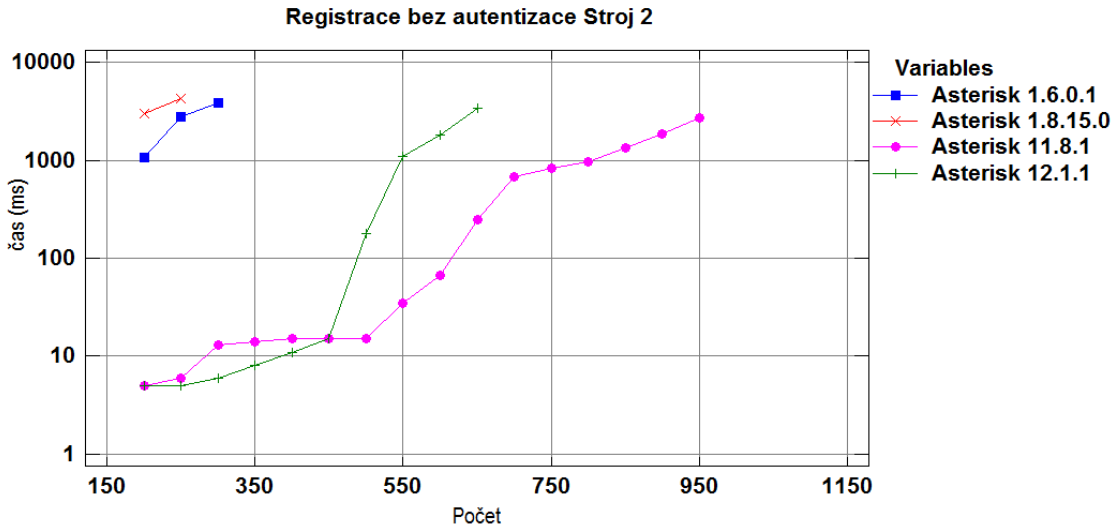
6.4 Časy REGISTER bez autentizace

Posledním scénářem, u něhož byl měřen čas jako hlavní parametr, je scénář *REGISTER* bez autentizace. Jako pomocným vodítkem při hledání dvou událostí, mezi kterými bude čas měřen, byl znovu standard RFC 6076. Takto byly nakonec vybrány události *REGISTER* a *200 OK* neboli časy úspěšné registrace.



Obr. 6.7: *Porovnání REGISTER bez autentizace u Stroje 3*

U stroje se slabším hardwarovým vybavením jde jednoznačně vidět rozdíly mezi jednotlivými verzemi Asterisku. Obzvláště Asterisk 11.8.1 mezi verzemi vyniká a dokáže zpracovat více současných hovorů.



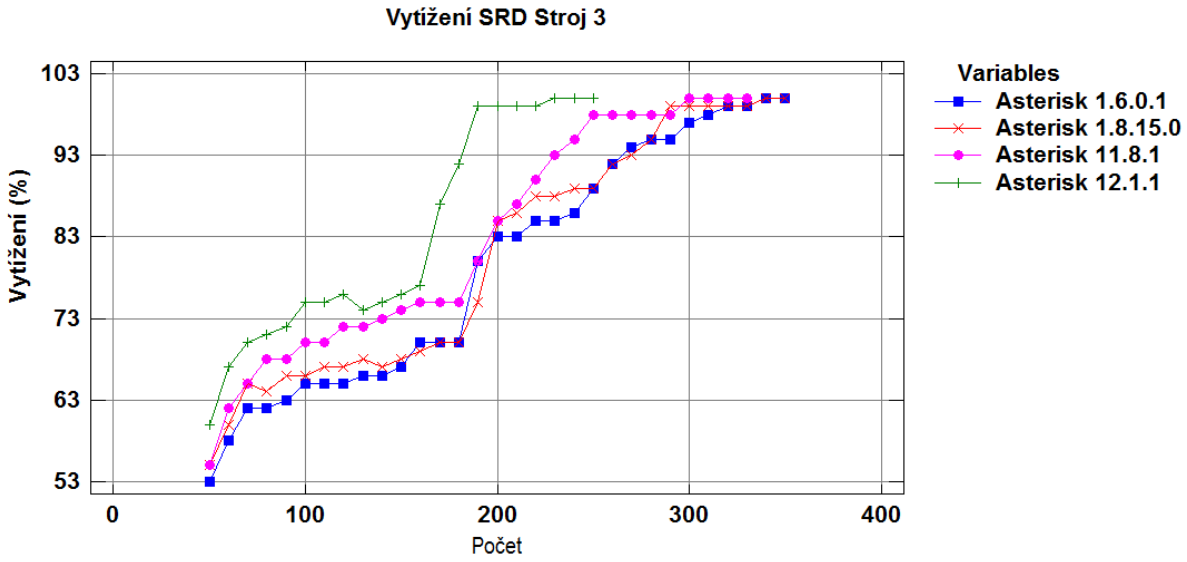
Obr. 6.8: *Porovnání REGISTER bez autentizace u Stroje 2*

Situace podobná jako u RRD nastává i u registrace bez autentizace. Žádost *REGISTER* způsobuje velké rozdíly mezi jednotlivými verzemi. Nejstarší verze 1.6.0.1 a 1.8.15.0 mají časy jednotlivých odpovědí daleko vyšší než novější verze.

6.5 Využití procesoru

Samozřejmě nejen časy ovlivňují výkon ústředny. Mezi další důležité parametry patří také využití procesoru. Procesor má výrazný vliv právě na jednotlivé časy odpovědí. Během měření byla jasně prokázána přímá úměrnost mezičasy odpovědí a využitím procesoru. Rozdíly se nejvýrazněji projevovaly v závislosti na tom, jaký stroj byl použit.

U výkonnostně slabšího stroje, Stroje 3, byl jednoznačně prokazatelný vliv počtu volání na výkonost ústředny. Jakmile se zvedl počet volání, tak stejně se zvýšil i výkon procesoru. A jakmile stoupnul výkon procesoru na 100%, tak se okamžitě zvýšil i čas, za který byl schopna ústředna odpovědět. Pro příklad bude uveden jeden graf využití procesoru.



Obr. 6.9: Porovnání vytížení procesoru u SRD Stroje 3

Při porovnání obrázků 6.10 a 6.1 je jasně patrné, že když se vytížení procesoru dostane na hodnotu 100% tak okamžitě stoupne i hodnota časů.

U výkonnostně silnějšího stroje, který má 8 procesorových jader, jsou patrné obrovské rozdíly mezi jednotlivými verzemi Asterisku. Asterisk 1.6.0.1 dokáže využít každé jádro jen z 35 – 40 %. Okamžitě při dosažení této hodnoty vytížení se začne projevovat zvětšení jednotlivých časů odpovědí. Asterisk 1.8.15.0 si naopak s více jádry neumí vůbec poradit. Při vyšším zatížení začne tato verze přenášet veškerou zátěž na jedno jádro a tím má tato verze jednoznačně nejhorší výsledky při testech jednotlivých scénářů. Asterisk 11.8.1 dokáže nejlépe ze všech testovaných verzí rozložit jednotlivou zátěž mezi jednotlivá jádra. I přesto nedokáže využít všechna jádra ze sta procent a tím si nakonec mírně zhoršuje vlastnosti. Poslední

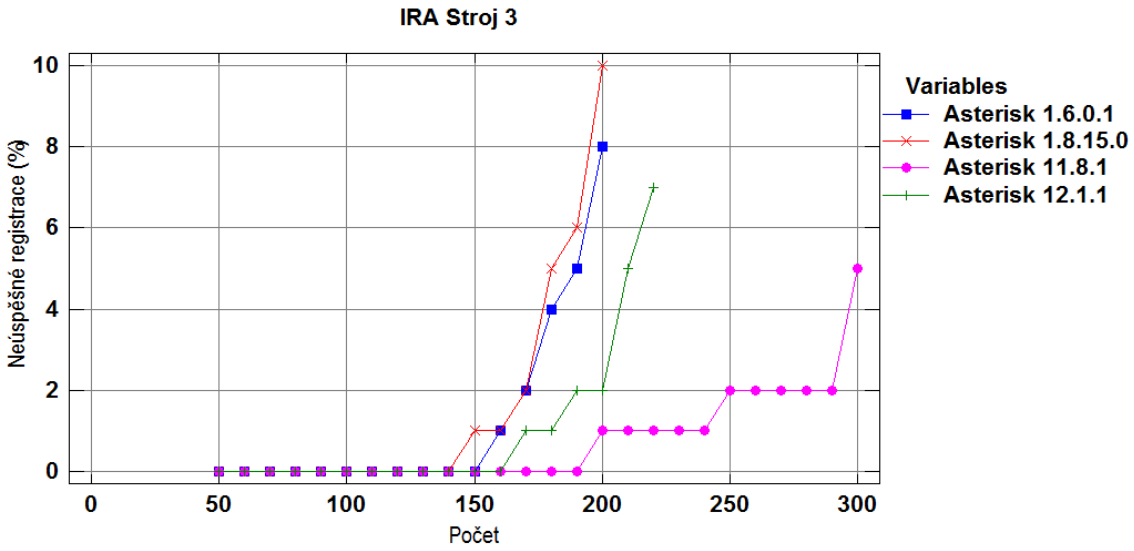
testovaná verze 12.1.1 sice umí dobře přenést zátěž mezi jednotlivá jádra, ovšem má nejvyšší časy odpovědí i při nejmenší zátěži, proto se toto dobré využívání nakonec ztrácí při vyšších počtu volání za sekundu.

6.6 Využití paměti

Spolu se zkoumáním vlivu procesoru na časy odpovědí, tato práce zjišťovala i vliv paměti na časy odpovědí. Vliv paměti na časy odpovědí nakonec prokázán nebyl. Ač se jakkoliv zvedla zátěž při kterékoliv verzi, tak paměť nikdy neprokazovala žádné zvýšení do kritických hodnot, čímž by ovlivňovala jednotlivé časy. Paměť dosahovala maximálně dvou třetin své hodnoty, přičemž hodnoty vytížení procesoru už dávno byly na 100%.

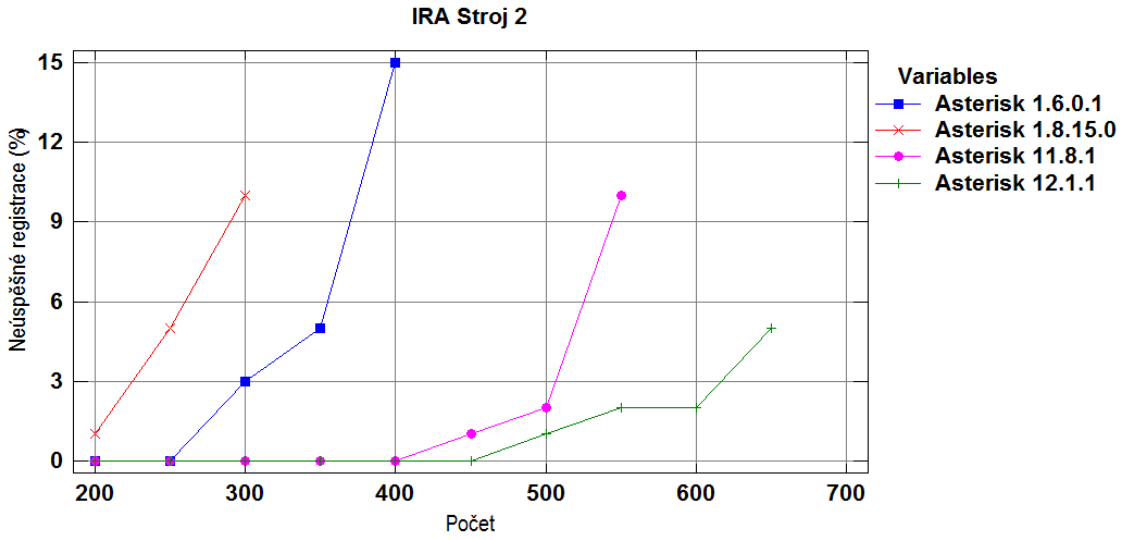
6.7 IRA

Spolu s měřením časů u jednotlivých odpovědí, byly také zaznamenávány neúspěšné pokusy o registraci. Toto měření bylo prováděno u scénáře s měřením času RRD. Počet neúspěšných registrací (popsáno v kapitole 2.2.2)je kritickou hodnotou při každém testování. Pokud je počet neúspěšných registrací příliš velký, tak to značí velkou zátěž ústředny, a s tím samozřejmě souvisí to, že ústředna poté není schopna zvládat další hovory a tak klesá využitelnost ústředny.



Obr. 6.10: *Porovnání IRA pro Stroj 3*

Z obrázku je patrné, že nejrychleji stoupá počet neúspěšných registrací u dvou nejstarších verzí, přičemž Asterisk 11.8.1 zaznamenává počet kritických neúspěchů až nad hodnotou více jak 250 hovorů za sekundu.

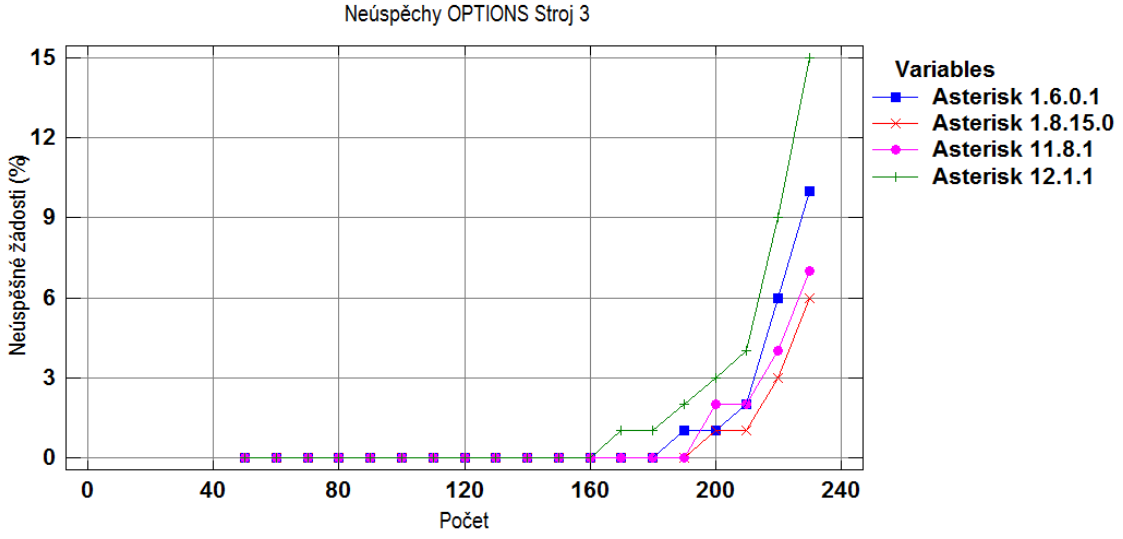


Obr. 6.11: *Porovnání IRA u Stroje 2*

Z porovnání neúspěšných aktualizací u výkonnostně lépe vybaveného stroje je možné vidět, že nejlépe si vedl Asterisk 12.1.1. Kdežto obě nejstarší verze Asterisku opět skončily v testu nejhůře.

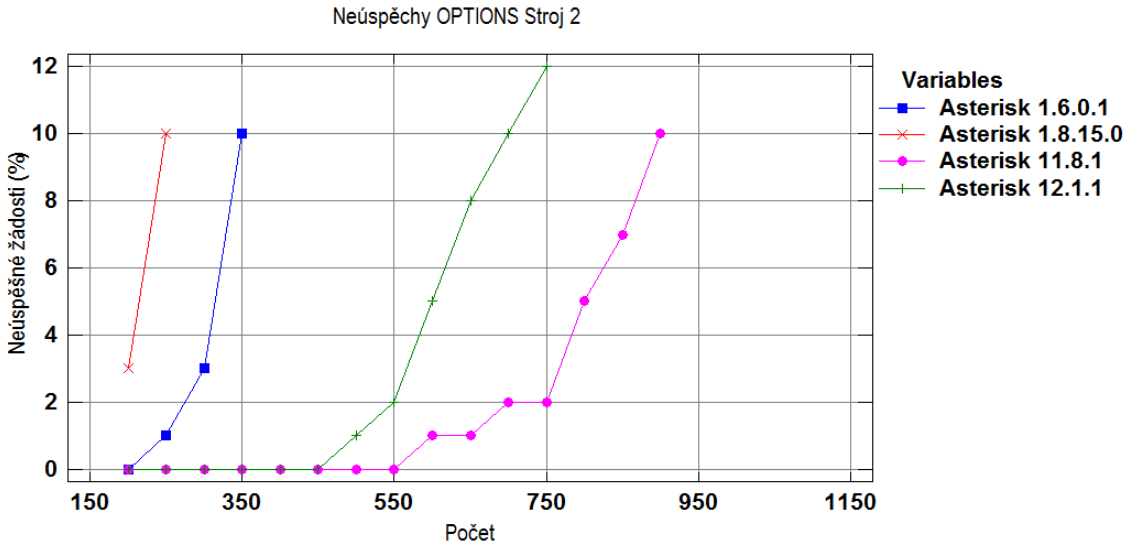
6.8 Neúspěšné žádosti OPTIONS

Posledním sledovaným scénářem byl scénář se žádostí *OPTIONS*. U tohoto scénáře byla pozornost zaměřena na kladné odpovědi na žádost, nikoliv na čas odpovědi. Měřením kladných odpovědí na žádost je možné pozorovat analogii vůči metrice IRA. Opět se jedná o procentuální vyjádření jednotlivých neúspěchů.



Obr. 6.12: *Porovnání neúspěchů OPTIONS u Stroje 3*

Z grafu lze pozorovat největší procento neúspěšných žádostí u verze Asterisk 12.1.1. Ostatní tři verze byly na tom velice podobně, co se týče procentního neúspěchu.



Obr. 6.13: *Porovnání neúspěchů OPTIONS u stroje 2*

Obrázek naznačuje opětovné problémy dvou verzí hned s velmi nízkými hodnotami počtu volání. Oproti tomu hlavně Asterisk 11.8.1 si v tomto ohledu vedl velice dobře a ke kritické hranici neúspěšných žádostí se dostal až velice vysokým počtem volání za sekundu.

6.9 Celkové zhodnocení verzí

Po popsání jednotlivých měření je potřebné shrnout jednotlivé výsledky do souhrnných informací a zhodnotit výsledky všech verzí s popisem toho, jaká verze je nejužitečnější k použití na kterém stroji.

6.9.1 Zhodnocení Asterisk 1.6.0.1

Ačkoliv se jedná o nejstarší testovanou verzi Asterisku, tak v testech byla verze 1.6.0.1 poměrně úspěšná. V testech na výkonnostně slabším stroji při testech pouze se žádostí *INVITE* příliš nezaostávala a efektivně dokázala zvládnout okolo 260 hovorů za sekundu. Naopak registrace dělal této verzi problémy, kdy se kritickou hranicí stala hodnota 160 hovorů za sekundu. Na výkonnostně silnějším stroji jednoznačně verze zaostává díky ne zcela dobře zvládnutému využití jader procesorů, což je asi největší slabinou této verze spolu s poměrně velkými hodnota neúspěšných registrací a žádostí *OPTIONS*.

6.9.2 Zhodnocení Asterisk 1.8.15.0

Verze 1.8.15.0 je dle testování plně koncipovaná na jednojádrové procesory. U testů Stroje 3 byla tato verze srovnatelná s novějšími verzemi, pokud se jednalo pouze o žádosti *INVITE*. Jakmile byla ve scénáři i žádost o registraci, tak výkon ústředny prudce klesal. Celkově

se dá verze poměrně srovnávat s verzí 1.6.0.1, neboť ve většině testů dosáhla srovnatelných výsledků. Ovšem co se týče vícejádrových procesorů, tak je tato verze absolutně nepoužitelná. Její výsledky se v takovýchto případech nedají ani hodnotit, protože u srovnatelného počtu volání za sekundu s jednojádrovým procesorem má skoro stejné výsledky. Asterisk 1.8.15.0 vytíží maximálně jen jedno jádro a s ním pracuje nehledě na další absolutně nevyužitá jádra.

6.9.3 Zhodnocení Asterisk 11.8.1

Nejlepší verzí Asterisku je podle testů verze 11.8.1. Verze pracuje jak s jedním, tak i více jádry nejlépe a nejlépe tak dokáže rozložit celou zátěž. Asterisk 11.8.1 dokáže efektivně zvládnout až 800 hovorů (kromě scénáře RRD) na Stroji 2 a 300 hovorů u stroje 3.

6.9.4 Zhodnocení Asterisk 12.1.1

Nejnovější z testovaných verzí Asterisku 12.1.1 měla jedny z nejhorších výsledků na slabším stroji, kde často byla horší i oproti nejstarším testovaným verzím. Naopak u vícejádrového procesoru naplno využila to, že umí rozložit zátěž mezi všechna jádra. Přesto se zátěž nedokázala dokonale rozložit a v porovnání s Asteriskem 11.8.1 jsou časy odpovědí horší i při nižší zátěži.

7 Závěr

Cílem práce bylo vytvořit generátor zátěže pro otestování SIP infrastruktury. Tento generátor měl být vytvořen za pomocí stávajících již existujících prvků. Zároveň měl tento generátor sloužit k otestování jednotlivých verzí B2BUA Asterisk. Momentálně již existuje několik generátorů, které slouží právě k testování SIP infrastruktury. Nicméně žádný z nich se nezaměřil na testování na základě RFC 6076 jako je to v případě této práce a scénářů zaměřených na standard RFC 6076.

Celá práce byla vytvořena jako kombinace několika volně dostupných prvků a skriptovacích jazyků. Skriptovací jazyk php spolu s knihovnou JpGraph a jazykem html umožnili vytvořit webové rozhraní, se kterým uživatel pracuje. Dále za pomoci jazyka Bash byly vytvořeny skripty, které zpracují přijatá data a také jsou takzvanou prodlouženou rukou jazyku php, díky které je možné spouštět či ukončovat aplikace i z webového rozhraní bez nutnosti zvýšených práv. Posledním využitým prvkem je program SIPp, který poskytnul generování zátěže. Po propojení všech věcí vzniknul komplexní systém, pomocí kterého byly důkladně otestovány čtyři verze Asterisku.

Díky webovému rozhraní již nebudou muset uživatelé zadávat parametry příkazu, který má být spuštěn, složitě do příkazové řádky. Stačí jim jednoduše otevřít internetový prohlížeč, ve kterém se jim zobrazí přehledně strukturovaný webový formulář pro zadávání dat. Pomocí jednoduchých kliknutí pak mohou získat nejen informace ohledně proběhnutého testu, ale také rovněž v přehledné grafické podobě informace o tom, kolik SIP událostí se vyskytlo v tom daném testu. Díky webovému rozhraní se výrazně také ulehčila práce se statistikami, které má uživatel prakticky ihned po testu k dispozici na místě, které si sám určí.

Prací byla rovněž testována možnost zaznamenávání dat pomocí paketového analyzátoru tcpdump, který zaznamenává cenná data o SIP přenosu. Ačkoliv jsou veškerá takto zaznamenaná data velice popisná, tak získávání důležitých informací z velikého kvanta nepotřebných zabere pak velkou porci času, díky kterému byl tento postup zavrhnut.

Po otestování všech čtyř verzí Asterisku na jednojádrovém i vícejádrovém procesoru se jako nejlepší jevila verze 11.8.1, která měla na obou testovaných strojích nejlepší jak časy odpovědí, tak procesorové využití. Na jednojádrovém procesoru nebyly rozdíly mezi jednotlivými verzemi až tak patrné, jak na vícejádrovém, kde nejstarší verze nedokázaly plně s více jádry spolupracovat a proto měly horší výsledky.

Práce by se dala jednoduchým způsobem rozšířit. Vzhledem k tomu, že momentálně jsou vytahovány jen opravdu důležité informace potřebné k otestování pomocí standardu RFC 6076, tak by bylo snadné tyto informace rozšířit a předat uživateli více informací. Práce by se dala rozšířit i formou získávání statistik z jiných zdrojů než za pomoci programu SIPp, ovšem musela by se najít schůdná cesta, díky které by později při velké zátěži nebylo získávání dat časově tak náročným úkonem.

Použitá literatura

[1] BANERJEE, K. SIP tutorial. [online]. [cit. 2014-03-21]. Dostupné z: http://www.siptutorial.net/SIP/

[2] SIP. [online]. [cit. 2014-03-21]. Dostupné z: https://sip.cesnet.cz/cs/protokoly/sip

[3] MALAS, Daryl. Basic Telephony SIP Performance Metrics. [online]. [cit. 2014-03-21]. Dostupné z: http://www.sipforum.org/component/option,com_docman/task,doc_view/gid,498/

[4] PUŽMANOVÁ, Rita. Protokol SIP ve zkratce. [online]. [cit. 2014-03-21]. Dostupné z: http://www.lupa.cz/clanky/protokol-sip-ve-zkratce/

[5] MALAS, Daryl a Al MORTON. Basic Telephony SIP End-to-End Performance Metrics: Request for Comments: 6076. [online]. [cit. 2014-03-21]. Dostupné z: http://datatracker.ietf.org/doc/rfc6076/?include_text=1

[6] SIP odpovědi. [online]. [cit. 2014-03-21]. Dostupné z: http://www.3cx.cz/voip-sip/sip-responses/

[7] VOZŇÁK, Miroslav. SIP/SDP. [online]. [cit. 2014-03-21]. Dostupné z: http://homel.vsb.cz/~voz29/voip/SIP.pdf

[8] VOZŇÁK, Miroslav. SIP/SDP. [online]. [cit. 2014-03-23]. Dostupné z: http://homel.vsb.cz/~voz29/files/VOIP_09.pdf

[9] VOZNAK,M., ROZHON, J. [I]Approach to stress tests in SIP environment based on marginal analysis.[/I] SPRINGER: Telecommunication Systems, 11p., 2013, ISSN 1018-4864.

[10] VOZNAK, M., ROZHON, J. [I]SIP Back to Back User Benchmarking.[/I] IEEE ICWMC 2010, pp. 92-96, 2010, Valencia, DOI 10.1109/ICWMC.2010.86

[11] GAYRAUD, Richard, Olivier JACQUES, Robert DAY, Charles P. WRIGHT. SIPp: SIPp reference documentation. [online]. [cit. 2014-03-21]. Dostupné z: http://sipp.sourceforge.net/doc3.3/reference.html

[12] StarTrinity SIP Tester™. [online]. [cit. 2014-03-21]. Dostupné z: http://startrinity.com/VoIP/SipTester/SipTester.aspx

[13] PacketGen™. [online]. [cit. 2014-03-21]. Dostupné z: http://www.gl.com/packetgen.html

[14] Asterisk cmd Originate. [online]. [cit. 2014-03-21]. Dostupné z: http://www.voip-info.org/wiki/view/Asterisk+cmd+Originate

[15] Asterisk Manager API Action Originate. [online]. [cit. 2014-03-21]. Dostupné z: http://www.voip-info.org/wiki/view/Asterisk+Manager+API+Action+Originate

[16] Asterisk cli originate. [online]. [cit. 2014-03-21]. Dostupné z: http://www.voip-info.org/wiki/view/Asterisk+cli+originate

[17] JOHNSTON, Alan B. *SIP: understanding the session initiation protocol*. 3rd ed. Boston: Artech House, 2009. ISBN 978-1-60783-995-8.

[18] SULTAN, Philippe. SIP peers external authentication in Asterisk / OpenPBX. [online]. [cit. 2014-03-23]. Dostupné z: https://who.rocq.inria.fr/Philippe.Sultan/Asterisk/asterisk_sip_external_authentication.html

[19] G.711. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2014 [cit. 2014-04-21]. Dostupné z: http://en.wikipedia.org/wiki/G.711

[20] FEMMINELLA, M., R. FRANCESCANGELI, F. GIACINTI, E. MACCHERANI, A. PARISI a G. REALI. Design, Implementation, and Performance Evaluation of an Advanced SIP-Based Call Control for VoIP Services. *2009 IEEE International Conference on Communications* [online]. IEEE, 2009, s. 1-5 [cit. 2014-04-29]. DOI: 10.1109/ICC.2009.5198905. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5198905

[21] KULIN, Merima, Tarik KAZAZ, Sasa MRDOVIC, E. MACCHERANI, A. PARISI a G. REALI. SIP server security with TLS: Relative performance evaluation. *2012 IX International Symposium on Telecommunications (BIHTEL)* [online]. IEEE, 2012, s. 1-6 [cit. 2014-04-29]. DOI: 10.1109/BIHTEL.2012.6412062. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6412062

Seznam příloh

Tištěné přílohy

Obsah CD

Složka asterisk

- sip.conf, extensions.conf – soubory pro konfiguraci Asterisku

Složka fonty

- fonty potřebné pro spolupráci s knihovnou JpGraph

Složka naměřené hodnoty

- soubory s naměřenými hodnotami

Složka script

- složka s Bash skripty

Složka sipp

- upravený sipp pro potřeby lepšího rtp přenosu

Složka www

- soubory pro běh webového rozhraní

Instalace.txt – textový soubor s pokyny pro instalaci generátoru

Konfigurace Asterisku.txt – textový soubor s konfigurací strojů s Asteriskem

Příkazy.txt – textový soubor se spouštěnými příkazy SIPp

SRD Stroj 2

Počet volání za sekundu	Asterisk 1.6.0.1 čas (ms)	Asterisk 1.8.15.0 čas (ms)	Asterisk 11.8.1 čas (ms)	Asterisk 12.1.1 čas (ms)
200	3	15	11	4
250	4	1115	12	5
300	6	1821	13	7
350	9	2886	13	11
400	10		15	19
450	14		16	20
500	19		18	22
550	74		18	237
600	273		72	506
650	2401		84	1211
700	3440		235	2723
750			360	
800			433	
850			707	
900			802	
950			1403	
1000			4315	

SRD Stroj 3

Počet volání za sekundu	Asterisk 1.6.0.1 čas (ms) / vytížení (%)	Asterisk 1.8.15.0 čas (ms) / vytížení (%)	Asterisk 11.8.1 čas (ms) / vytížení (%)	Asterisk 12.1.1 čas (ms) / vytížení (%)
50	2 / 53	3 / 55	4 / 55	7 / 60
70	3 / 62	4 / 65	6 / 65	13 / 70
90	5 / 63	6 / 66	9 / 68	17 / 72
110	7 / 65	9 / 67	13 / 70	22 / 75
130	9 / 66	12 / 68	16 / 72	44 / 74
150	12 / 67	17 / 68	22 / 74	73 / 76
170	16 / 70	20 / 70	24 / 75	181 / 87
190	20 / 80	23 / 75	78 / 80	260 / 99
210	25 / 83	33 / 86	125 / 87	252 / 99
230	28 / 85	59 / 88	237 / 93	311 / 100
250	35 / 89	147 / 89	226 / 98	3805 / 100
270	92 / 94	176 / 93	234 / 98	
290	263 / 95	197 / 99	327 / 98	
300	232 / 97	215 / 99	238 / 100	
310	252 / 98	206 / 99	243 / 100	
320	299 / 99	247 / 99	269 / 100	
330	333 / 99	303 / 99	2006 / 100	
340	460 / 100	261 / 100		
350	2534 / 100	2240 / 100		

RRD Stroj 2

Počet volání za sekundu	Asterisk 1.6.0.1 čas (ms) / IRA (%)	Asterisk 1.8.15.0 čas (ms) / IRA (%)	Asterisk 11.8.1 čas (ms) / IRA (%)	Asterisk 12.1.1 čas (ms) / IRA (%)
200	37 / 0	3616 / 1	5 / 0	4 / 0
250	1996 / 0	5286 / 5	25 / 0	5 / 0
300	3476 / 3	6142 / 10	70 / 0	6 / 0
350	4089 / 5		86 / 0	12 / 0
400	5123 / 15		103 / 0	19 / 0
450			2027 / 1	599 / 0
500			4021 / 2	1422 / 1
550			4801 / 10	3015 / 2
600				3842 / 2
650				4361 / 5

RRD Stroj 3

Počet volání za sekundu	Asterisk 1.6.0.1 čas (ms) / vytížení (%) / IRA (%)	Asterisk 1.8.15.0 čas (ms) / vytížení (%) / IRA (%)	Asterisk 11.8.1 čas (ms) / vytížení (%) / IRA (%)	Asterisk 12.1.1 čas (ms) / vytížení (%) / IRA (%)
50	4 / 53 / 0	8 / 62 / 0	5 / 60 / 0	10 / 70 / 0
70	9 / 60 / 0	14 / 73 / 0	11 / 70 / 0	15 / 75 / 0
90	12 / 70 / 0	29 / 75 / 0	22 / 70 / 0	30 / 78 / 0
110	18 / 78 / 0	41 / 80 / 0	22 / 70 / 0	33 / 80 / 0
130	22 / 85 / 0	49 / 83 / 0	30 / 72 / 0	39 / 78 / 0
150	29 / 85 / 0	75 / 92 / 1	49 / 76 / 0	71 / 82 / 0
170	39 / 88 / 2	101 / 95 / 2	64 / 80 / 0	149 / 93 / 1
190	786 / 98 / 5	475 / 100 / 6	78 / 87 / 0	206 / 99 / 2
200	1703 / 99 / 8	1057 / 100 / 10	145 / 95 / 1	507 / 100 / 2
210			166 / 95 / 1	707 / 100 / 5
220			201 / 95 / 1	3127 / 100 / 7
240			245 / 98 / 1	
250			262 / 98 / 2	
260			268 / 99 / 2	
270			284 / 99 / 2	
280			420 / 100 / 2	
290			470 / 100 / 2	
300			1035 / 100 / 5	

Registrace bez autentizace Stroj 2

Počet volání za sekundu	Asterisk 1.6.0.1 čas (ms)	Asterisk 1.8.15.0 čas (ms)	Asterisk 11.8.1 čas (ms)	Asterisk 12.1.1 čas (ms)
200	1075	3030	5	5
250	2758	4317	6	5
300	3876		13	6
350			14	8
400			15	11
450			15	15
500			15	180
550			35	1106
600			66	1824
650			250	3421
700			680	
750			829	
800			976	
850			1347	
900			1847	
950			2684	

Registrace bez autentizace Stroj 3

Počet volání za sekundu	Asterisk 1.6.0.1 čas (ms) / vytížení (%)	Asterisk 1.8.15.0 čas (ms) / vytížení (%)	Asterisk 11.8.1 čas (ms) / vytížení (%)	Asterisk 12.1.1 čas (ms) / vytížení (%)
50	3 / 68	4 / 71	3 / 60	9 / 70
70	4 / 72	9 / 73	7 / 67	14 / 75
90	6 / 75	20 / 75	15 / 70	22 / 75
110	9 / 77	26 / 78	19 / 70	22 / 75
130	11 / 85	31 / 88	20 / 75	26 / 76
150	203 / 98	61 / 92	45 / 75	35 / 84
160	268 / 98	77 / 93	43 / 75	39 / 90
170	488 / 98	77 / 95	59 / 80	112 / 94
180	1851 / 100	99 / 100	59 / 82	115 / 100
190		748 / 100	62 / 90	185 / 100
200		2386 / 100	93 / 92	178 / 100
210			110 / 94	249 / 100
220			142 / 96	322 / 100
230			184 / 96	2859 / 100
250			190 / 97	
270			225 / 97	
280			260 / 99	
290			2165 / 100	

Options Stroj 2

Počet volání za sekundu	Asterisk 1.6.0.1 počet neúspěšných žádostí (%)	Asterisk 1.8.15.0 počet neúspěšných žádostí (%)	Asterisk 11.8.1 počet neúspěšných žádostí (%)	Asterisk 12.1.1 počet neúspěšných žádostí (%)
200	0	3	0	0
250	1	10	0	0
300	3		0	0
350	10		0	0
400			0	0
450			0	0
500			0	1
550			0	2
600			1	5
650			1	8
700			2	10
750			2	12
800			5	
850			7	
900			10	

Options Stroj 3

Počet volání za sekundu	Asterisk 1.6.0.1 počet neúspěšných žádostí (%)	Asterisk 1.8.15.0 počet neúspěšných žádostí (%)	Asterisk 11.8.1 počet neúspěšných žádostí (%)	Asterisk 12.1.1 počet neúspěšných žádostí (%)
50	0	0	0	0
70	0	0	0	0
80	0	0	0	0
90	0	0	0	0
100	0	0	0	0
110	0	0	0	0
120	0	0	0	0
130	0	0	0	0
140	0	0	0	0
150	0	0	0	0
160	0	0	0	0
170	0	0	0	1
180	0	0	0	1
190	1	0	0	2
200	1	1	2	3
210	2	1	2	4
220	6	3	4	9
230	10	6	7	15

INVITE s autentizací Stroj 2

Počet volání za sekundu	Asterisk 1.6.0.1 čas (ms)	Asterisk 1.8.15.0 čas (ms)	Asterisk 11.8.1 čas (ms)	Asterisk 12.1.1 čas (ms)
200	5	1444	4	10
250	8	2473	5	11
300	9	3360	6	12
350	23		8	17
400	29		12	248
450	32		14	264
500	39		18	302
550	293		21	412
600	1645		25	1554
650	3048		258	2228
700	4215		1100	3152
750			2121	
800			3868	

INVITE s autentizací Stroj 3

Počet volání za sekundu	Asterisk 1.6.0.1 čas (ms) / vytížení (%)	Asterisk 1.8.15.0 čas (ms) / vytížení (%)	Asterisk 11.8.1 čas (ms) / vytížení (%)	Asterisk 12.1.1 čas (ms) / vytížení (%)
50	8 / 33	6 / 35	8 / 45	7 / 50
70	14 / 50	9 / 43	14 / 60	20 / 68
90	20 / 60	15 / 60	22 / 70	33 / 82
110	31 / 75	20 / 70	32 / 78	58 / 85
130	49 / 80	26 / 77	52 / 83	104 / 88
150	62 / 85	40 / 81	72 / 85	123 / 90
170	70 / 87	62 / 85	89 / 87	188 / 96
190	78 / 88	120 / 90	125 / 90	250 / 99
210	84 / 89	200 / 90	197 / 91	306 / 100
230	93 / 89	244 / 93	257 / 95	365 / 100
240	105 / 92	244 / 95	263 / 96	400 / 100
250	120 / 94	245 / 97	288 / 96	526 / 100
260	204 / 96	258 / 98	304 / 96	2147 / 100
280	240 / 98	303 / 99	340 / 99	
300	303 / 100	322 / 100	357 / 100	
310	345 / 100	2025 / 100	377 / 100	
320	1320 / 100		1722 / 100	